LA-UR-19-28621
Approved for public release; distribution is unlimited.

Title: Discretely Conservative Multiphase Advection on Unstructured Meshes Using the Interface Reconstruction Library

Author(s): Chiodi, Robert Michael

Intended for: Possible presentation at my home university, Cornell University.

Issued: 2019-08-26

Discretely Conservative Multiphase Advection on Unstructured Meshes Using the Interface Reconstruction Library

Robert Chiodi

Cornell University

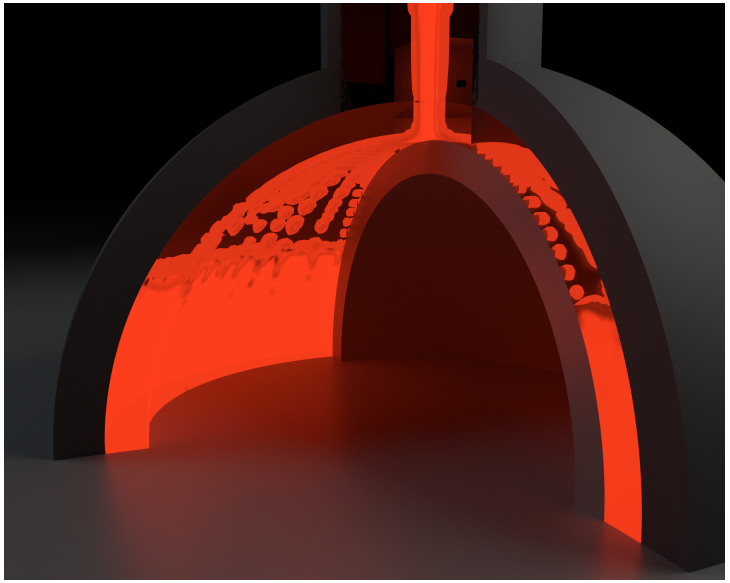
CCS-2 Summer Intern

Truchas

Simulates advanced metal casting processes of exotic materials

Incorporates heating of mold, injection of molten metal, and solidification.

Used as a design tool to adjust manufacturing processes.

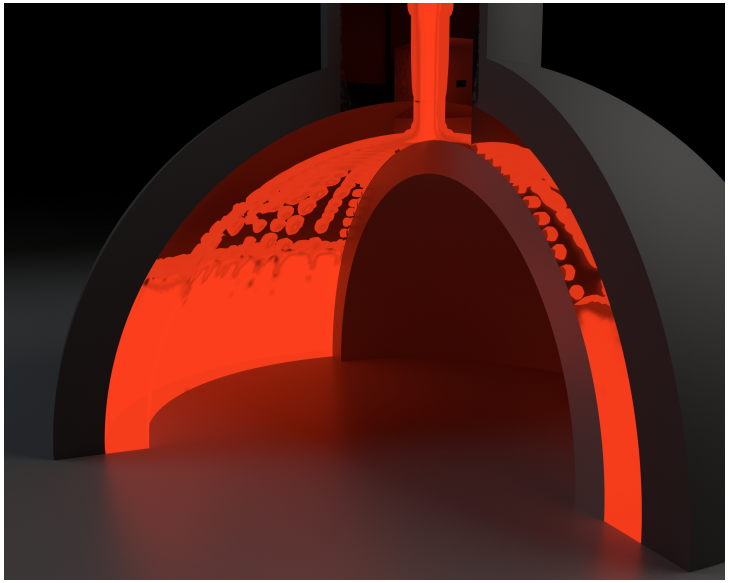


Truchas

Simulates advanced metal casting processes of exotic materials

Incorporates heating of mold, injection of molten metal, and solidification.

Used as a design tool to adjust manufacturing processes.



Truchas

Characteristics of Truchas Flows

- Complicated domains
- High density ratios between fluids
- Multiple materials in domain
- Multiple scalars to track (temperature, enthalpy, etc.)

Truchas

Characteristics of Truchas Flows

- Complicated domains
- High density ratios between fluids
- Multiple materials in domain
- Multiple scalars to track (temperature, enthalpy, etc.)

Numerical Requirements

Truchas

Characteristics of Truchas Flows

- Complicated domains
- High density ratios between fluids
- Multiple materials in domain
- Multiple scalars to track (temperature, enthalpy, etc.)

Numerical Requirements

- Ability to use unstructured meshes with heterogeneous element types

Truchas

Characteristics of Truchas Flows

- Complicated domains
- High density ratios between fluids
- Multiple materials in domain
- Multiple scalars to track (temperature, enthalpy, etc.)

Numerical Requirements

- Ability to use unstructured meshes with heterogeneous element types
- Robust momentum solver

Truchas

Characteristics of Truchas Flows

- Complicated domains
- High density ratios between fluids
- Multiple materials in domain
- Multiple scalars to track (temperature, enthalpy, etc.)

Numerical Requirements

- Ability to use unstructured meshes with heterogeneous element types
- Robust momentum solver
- Track multiple phases and represent multiple interfaces in one cell

Truchas

Characteristics of Truchas Flows

- Complicated domains
- High density ratios between fluids
- Multiple materials in domain
- Multiple scalars to track (temperature, enthalpy, etc.)

Numerical Requirements

- Ability to use unstructured meshes with heterogeneous element types
- Robust momentum solver
- Track multiple phases and represent multiple interfaces in one cell
- Consistent fluxing of volumes with associated scalars

Truchas

Characteristics of Truchas Flows

- Complicated domains
- High density ratios between fluids
- Multiple materials in domain
- Multiple scalars to track (temperature, enthalpy, etc.)

Numerical Requirements

- Ability to use unstructured meshes with heterogeneous element types
- Robust momentum solver
- Track multiple phases and represent multiple interfaces in one cell
- Consistent fluxing of volumes with associated scalars
- Need to have reasonable time-to-solution

Truchas

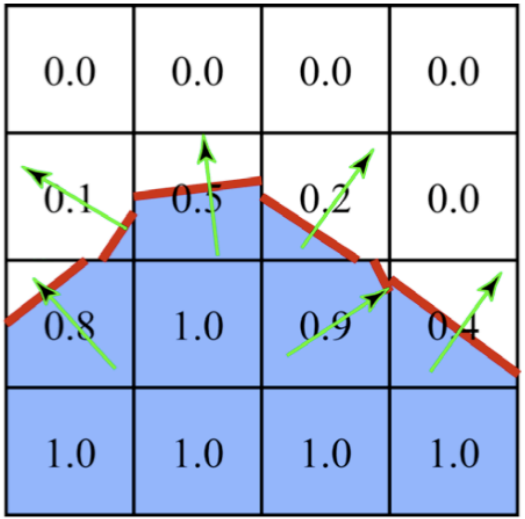
Numerical Requirements

- Ability to use unstructured meshes with heterogeneous element types
- Robust momentum solver
- Track multiple phases and represent multiple interfaces in one cell
- Consistent fluxing of volumes with associated scalars
- Need to have reasonable time-to-solution

Geometric Volume of Fluid Methods

- Work with general polyhedra
- Can couple phase volume fluxes with other fluxes (i.e. momentum, scalars)
- Multiple phases possible with onion-skin or nested-dissection
- Unconditionally stable, can use at high CFL

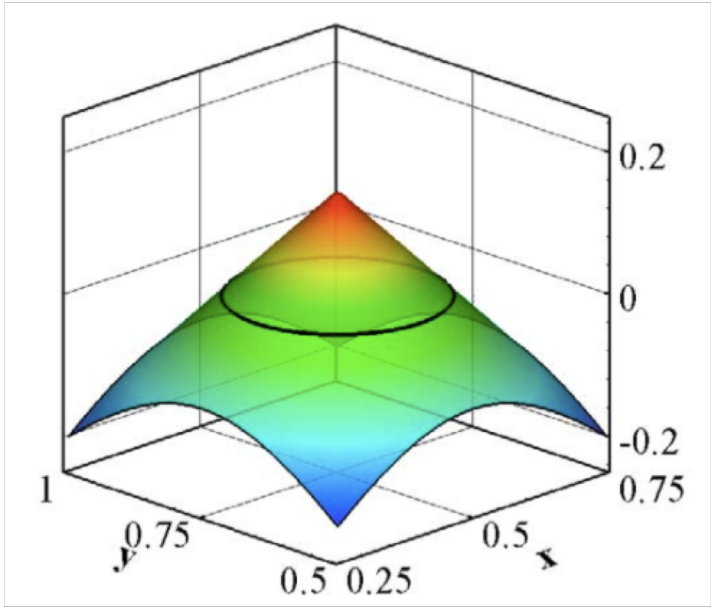
Geometric Volume of Fluid Methods



Volume of Fluid

Transport: Volume

Find: Interface

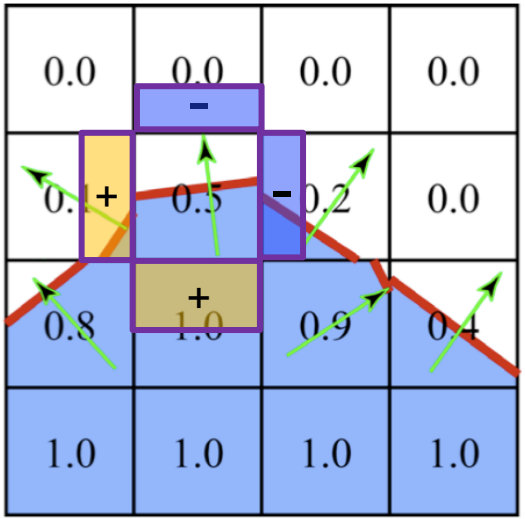


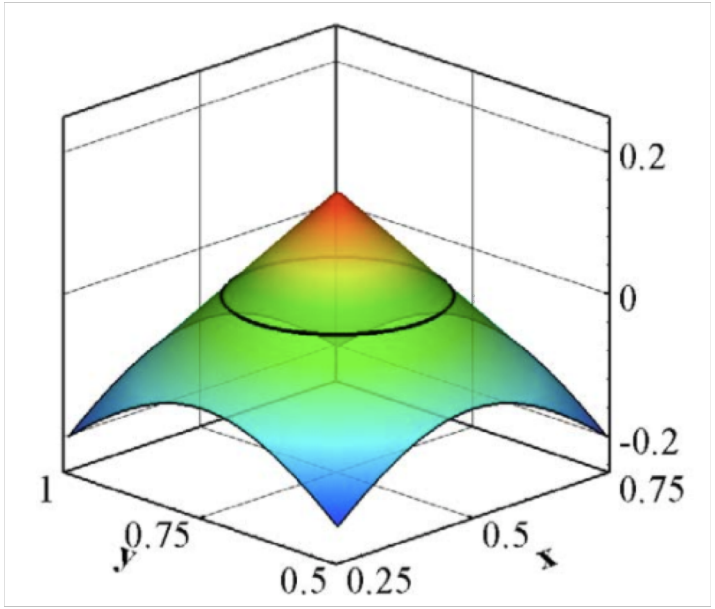
Level Set

Transport: Interface (implicitly)

Find: Volume

Geometric Volume of Fluid Methods





Volume of Fluid

Transport: Volume

Find: Interface

$$\alpha^{n+1} = \alpha^{n} + \frac{1}{V_c}\sum_{f} F_{\alpha}$$

Level Set

Transport: Interface (implicitly)

Find: Volume

Geometric Volume of Fluid Methods

Geometric Advection

- Formation of flux volumes
- Intersection with the underlying mesh
- Update of cell volume moments

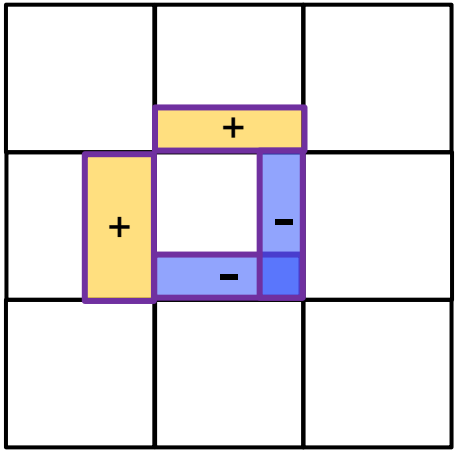
Interface Reconstruction

- Assume representation of the interface in each cell
- Orient interface representation from transported quantities
- Enforce reconstruction recovers fraction of each phase in cell

Truchas – Old Method

$$f_{vol} = \Delta t \, A_f U_f$$

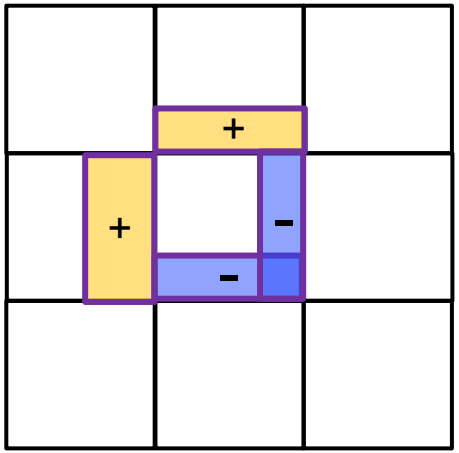
$$\alpha^{n+1} = \alpha^n + \frac{1}{V_c} \sum_f F_\alpha$$

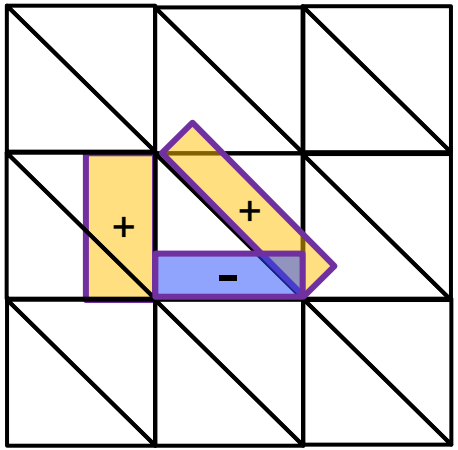


Fixes to reduce error:

1. Effectively lower time-step through using sub-steps
2. Iteratively modify fluxed material volumes to match total volume and preserve boundedness
3. Individually adjust cell face fluxes to preserve boundedness

Truchas – Old Method





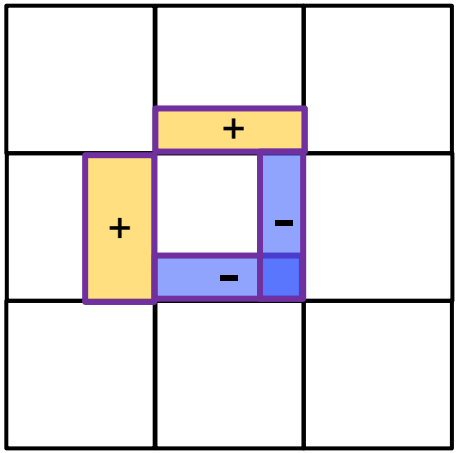
$$f_{vol} = \Delta t \, A_f U_f$$

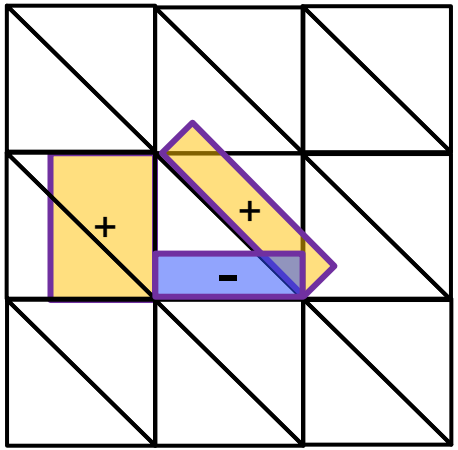
$$\alpha^{n+1} = \alpha^n + \frac{1}{V_c} \sum_f F_\alpha$$

Fixes to reduce error:

1. Effectively lower time-step through using sub-steps
2. Iteratively modify fluxed material volumes to match total volume and preserve boundedness
3. Individually adjust cell face fluxes to preserve boundedness

Truchas – Old Method





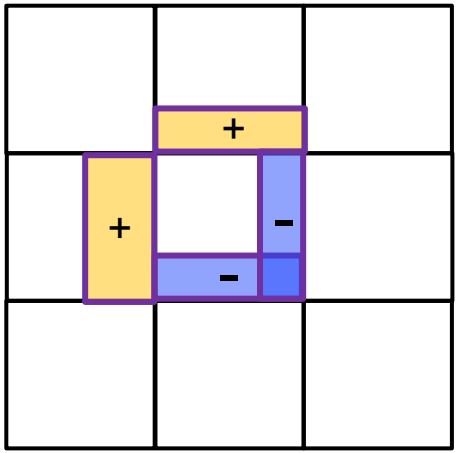
$$f_{vol} = \Delta t \, A_f U_f$$

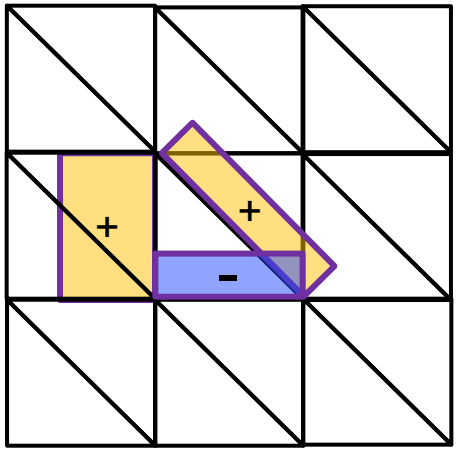
$$\alpha^{n+1} = \alpha^n + \frac{1}{V_c} \sum_f F_\alpha$$

Fixes to reduce error:

1. Effectively lower time-step through using sub-steps
2. Iteratively modify fluxed material volumes to match total volume and preserve boundedness
3. Individually adjust cell face fluxes to preserve boundedness

Truchas – Old Method





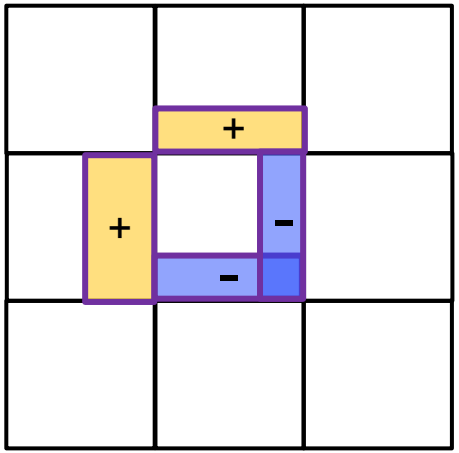
$$f_{vol} = \Delta t \, A_f U_f$$

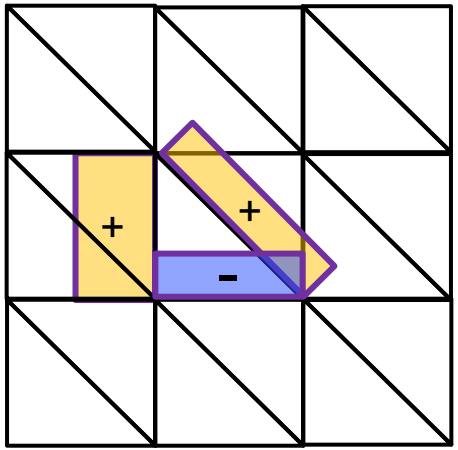
$$\alpha^{n+1} = \alpha^n + \frac{1}{V_c} \sum_f F_\alpha$$

Fixes to reduce error:

1. Effectively lower time-step through using sub-steps
2. Iteratively modify fluxed material volumes to match total volume and preserve boundedness
3. Individually adjust cell face fluxes to preserve boundedness

Truchas – Old Method





$$f_{vol} = \Delta t \, A_f U_f$$

$$\alpha^{n+1} = \alpha^n + \frac{1}{V_c} \sum_f F_\alpha$$

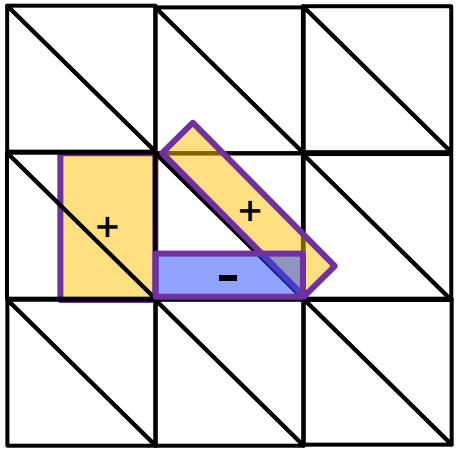
Fixes to reduce error:

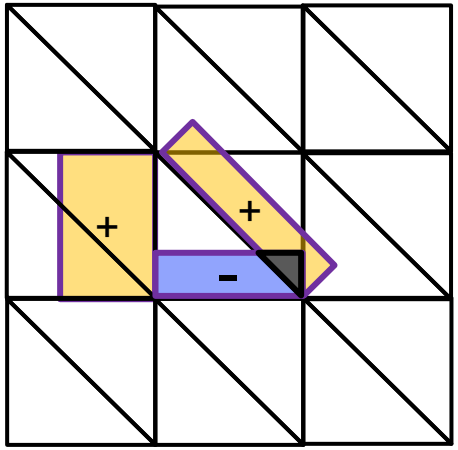
1. Effectively lower time-step through using sub-steps
2. Iteratively modify fluxed material volumes to match total volume and preserve boundedness
3. Individually adjust cell face fluxes to preserve boundedness

1. Ivey & Moin, JCP. 2017

Non-Intersecting Flux Polyhedron Advection[1]

$$f_{vol} = \Delta t \, A_f U_f$$





Central idea: Define priority order and do not allow volume to fluxed twice

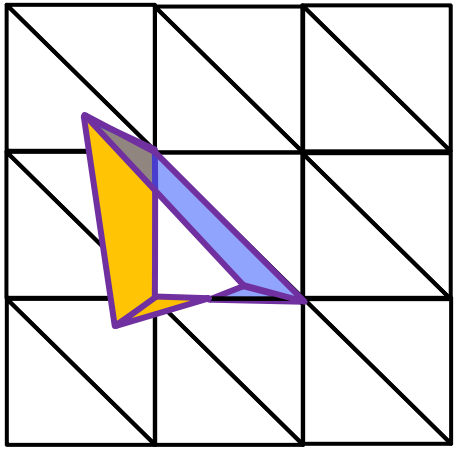
1. Iterates as Truchas does to match flux volumes
2. Must intersect face flux polyhedral to determine if volume already taken
3. Results become dependent on order of face fluxing

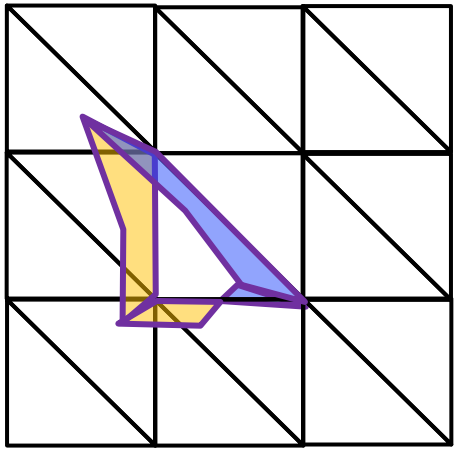
Edge-Matched Flux Polyhedron Advection

1. Ivey & Moin, JCP. 2017
2. Jofre et. al, C&F. 2014
3. Owkes & Desjardins, JCP. 2014
4. Comminal et. al, JCP. 2015

$$f_{vol} = \Delta t \, A_f U_f$$

$$\alpha^{n+1} = \alpha^n + \frac{1}{V_c} \sum_f F_\alpha$$





1. Use nodal velocities to project backwards in time and form flux polyhedron
2. Correct polyhedron to match expected volume flux

Geometric Volume of Fluid Methods

Geometric Advection

- Formation of flux volumes
- Intersection with the underlying mesh
- Update of cell volume moments

Interface Reconstruction

- Assume representation of the interface in each cell
- Orient interface representation from transported quantities
- Enforce reconstruction recovers fraction of each phase in cell

Geometric Volume of Fluid Methods

Geometric Advection

- Formation of flux volumes
- Intersection with the underlying mesh
- Update of cell volume moments

Interface Reconstruction

- Assume representation of the interface in each cell
- Orient interface representation from transported quantities
- Enforce reconstruction recovers fraction of each phase in cell

Geometric Volume of Fluid Methods

Geometric Advection

- Formation of flux volumes
- Intersection with the underlying mesh
- Update of cell volume moments

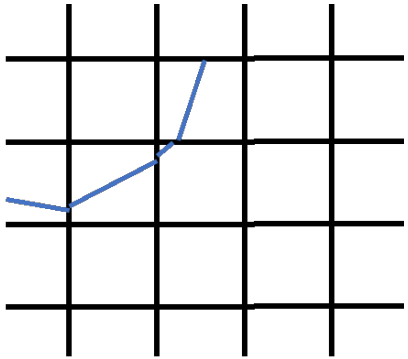
Complicated and expensive computational geometry problems

Interface Reconstruction

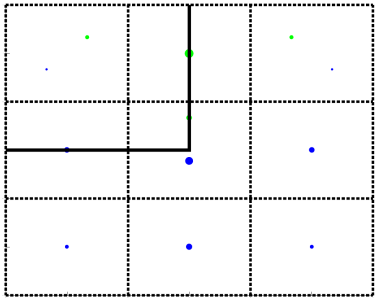
- Assume representation of the interface in each cell
- Orient interface representation from transported quantities
- Enforce reconstruction recovers fraction of each phase in cell

Interface Reconstruction Library (IRL)

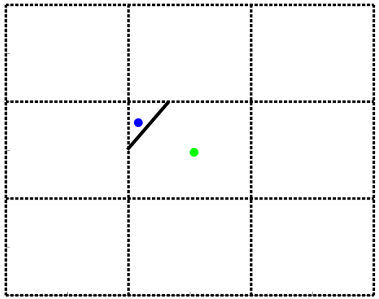
Goal: Enable wider adoption of geometric transport schemes and algorithms requiring complex computational geometry



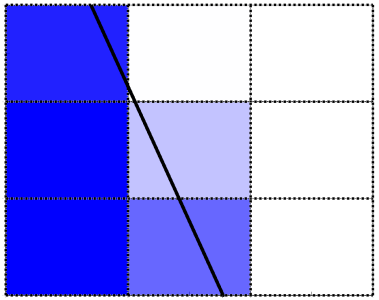
Geometric Advection



R2P Reconstruction



MoF Reconstruction



ELVIRA Reconstruction

Interface Reconstruction Library (IRL)

What do these operations require?

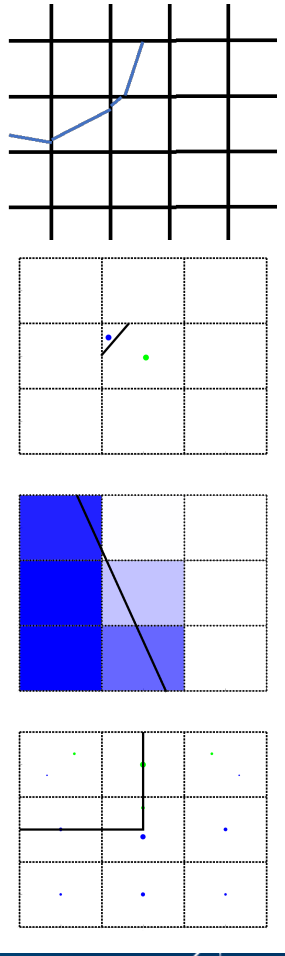
Obtaining some moments for the intersection between some volume and a space enclosed by some planes.

some volume
- A polyhedron or polygon
- ➢ Computational cells, flux volumes, bounding boxes, supercells,…

some planes
- A collection of n-planes in 3D that partition Euclidean space
- ➢ Cell boundaries, liquid-gas interfaces, arbitrary convex 2/3-polytopes,…

some moments
- Moments from integrating over the intersection of some volume and the space enclosed by some planes
- ➢ Volume, centroid, face normal vector,…

IRL provides high-performance, general algorithms usable for structured, unstructured, and AMR frameworks

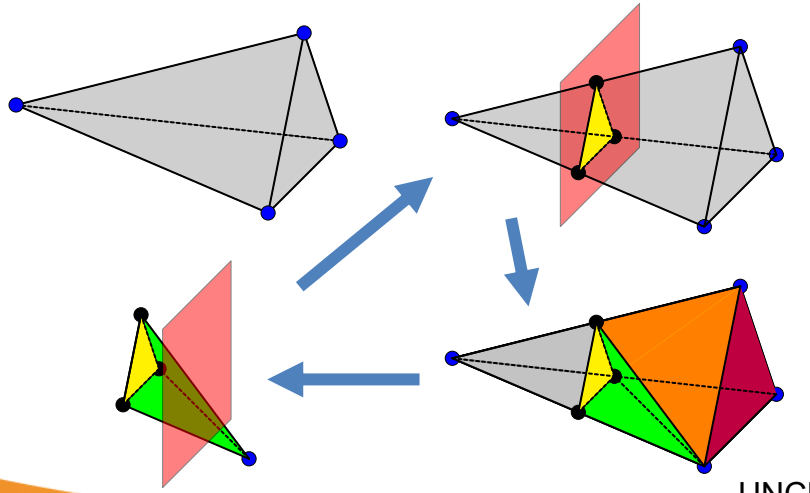


Interface Reconstruction Library (IRL)

Available Volume Intersection Methods

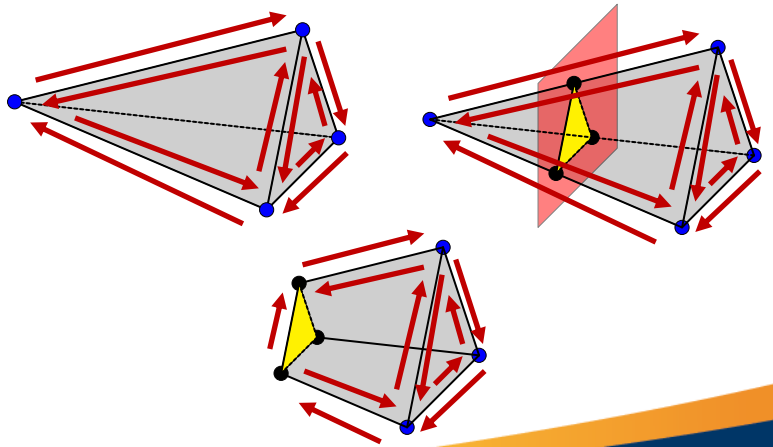
Recursive Simplex

1. Decompose a polytope into set of simplices
2. Intersect each simplex with the plane(s)
3. Sum up contribution to the volume moments



Half-Edge Clip & Cap

1. Transform polytope to half-edge structure
2. Clip & Cap half-edge structure
3. Compute volume moments through face triangulation

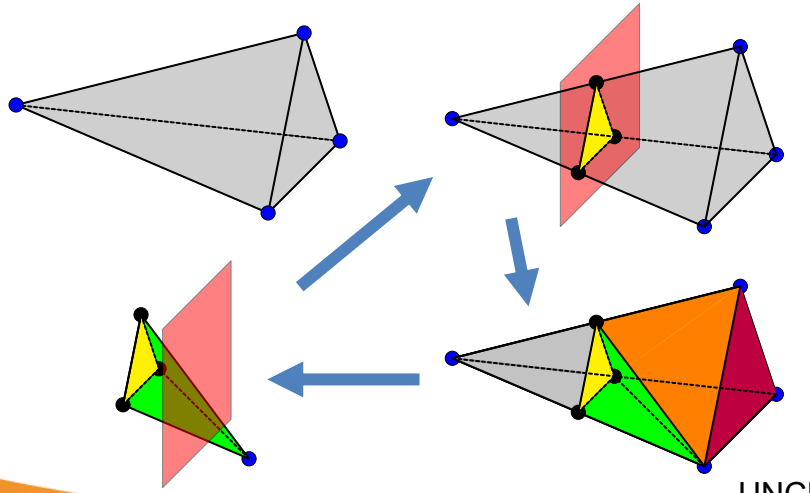


Interface Reconstruction Library (IRL)

Available Volume Intersection Methods

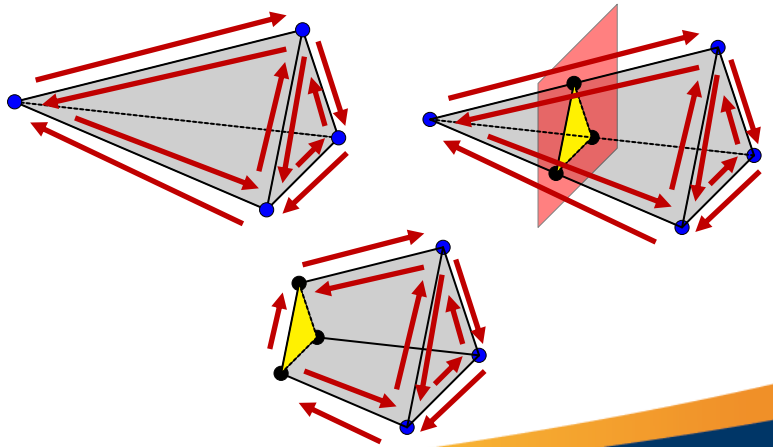
Recursive Simplex

1. Decompose a polytope into set of simplices
2. Intersect each simplex with the plane(s)
3. Sum up contribution to the volume moments

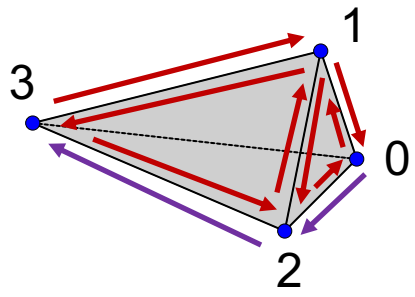


Half-Edge Clip & Cap

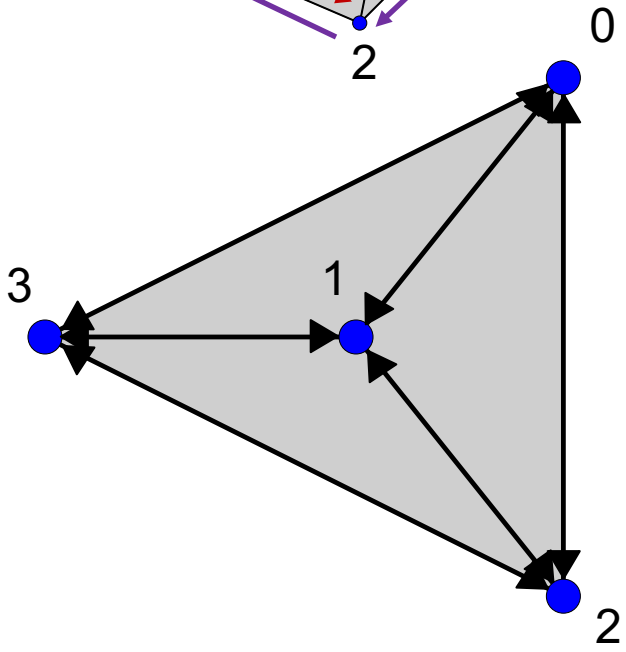
1. Transform polytope to half-edge structure
2. Clip & Cap half-edge structure
3. Compute volume moments through face triangulation



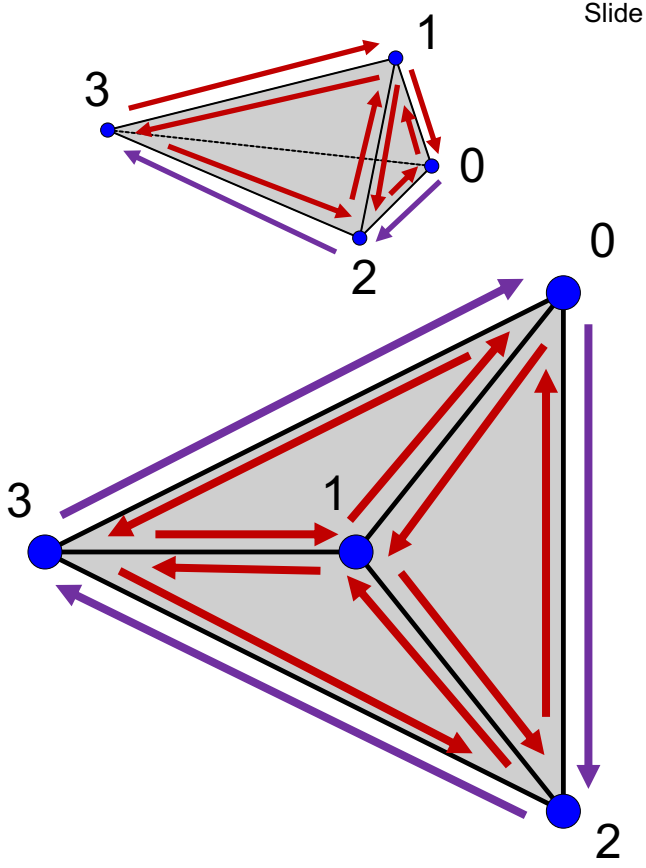
Half-Edge Structures



A typical representation of a polytope might include vertices, edges, and faces. The edges form a bi-directional undirected graph for the vertices



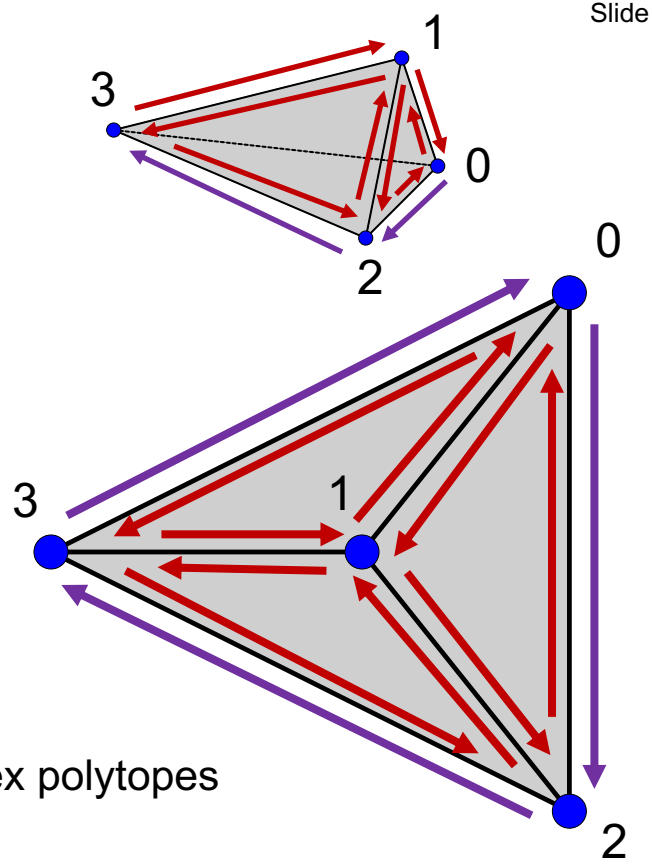
Half-Edge Structures



A typical representation of a polytope might include vertices, edges, and faces. The edges form a bi-directional undirected graph for the vertices.

A half edge structure represents the polytope with a directed graph of the vertices. Each edge is represented by two directed half edges.

Half-Edge Structures



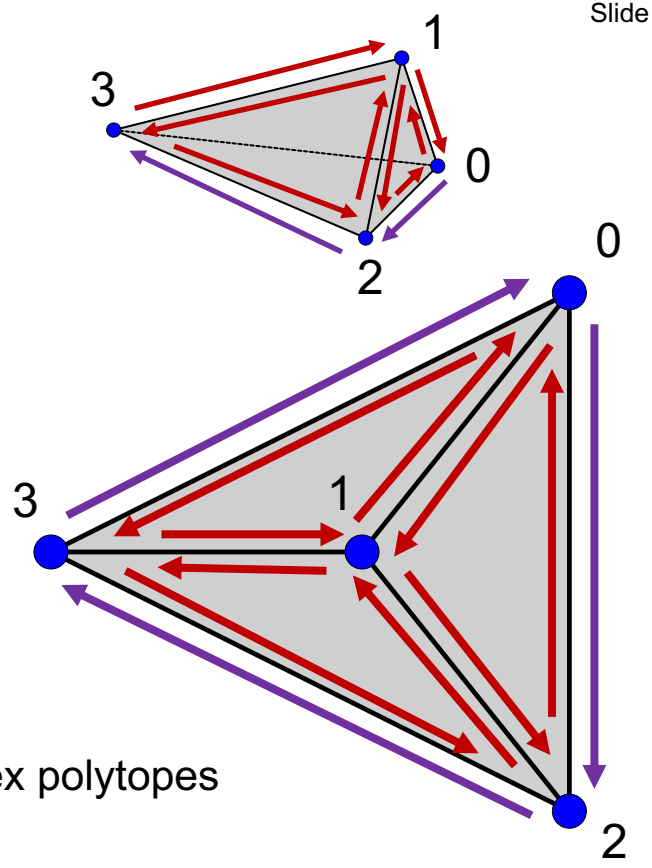
Important half-edge structure properties

1. Face consists of a close loop of half-edges
2. Each half-edge has a mirror half edge that exists on the face sharing that edge
3. Known traversal direction for the face

This provides

1. Easy traversal of faces, vertices, and edges
2. Ability to calculate moments for convex and non-convex polytopes
3. Increased efficiency for volume intersection operations

Half-Edge Structures



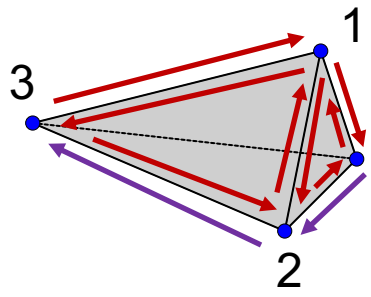
Important half-edge structure properties

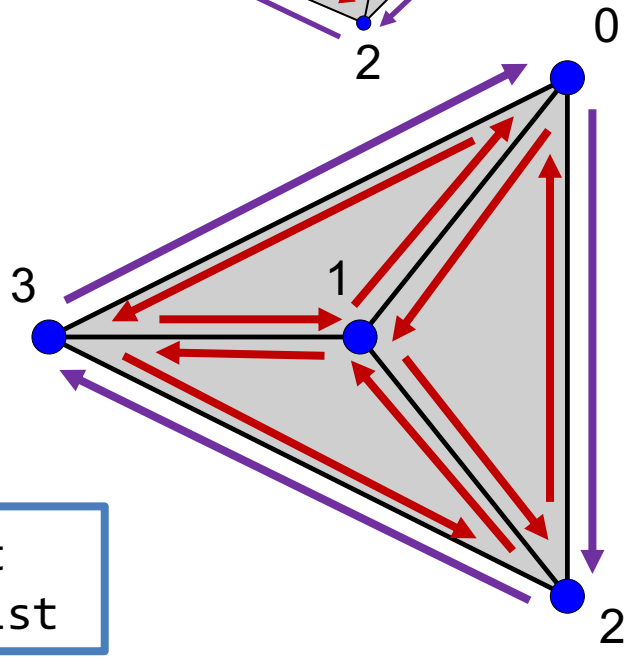
1. Face consists of a close loop of half-edges
2. Each half-edge has a mirror half edge that exists on the face sharing that edge
3. Known traversal direction for the face

This provides

1. Easy traversal of faces, vertices, and edges
2. Ability to calculate moments for convex and non-convex polytopes
3. Increased efficiency for volume intersection operations

Half-Edge Structures – Implementation





HalfEdge

```
HalfEdge* previous
HalfEdge* next
Face* my_face
Vertex* edge_end
```

Vertex

```
Pt location
HalfEdge* any_half_edge
```

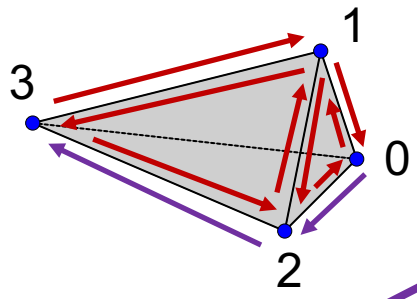
Face

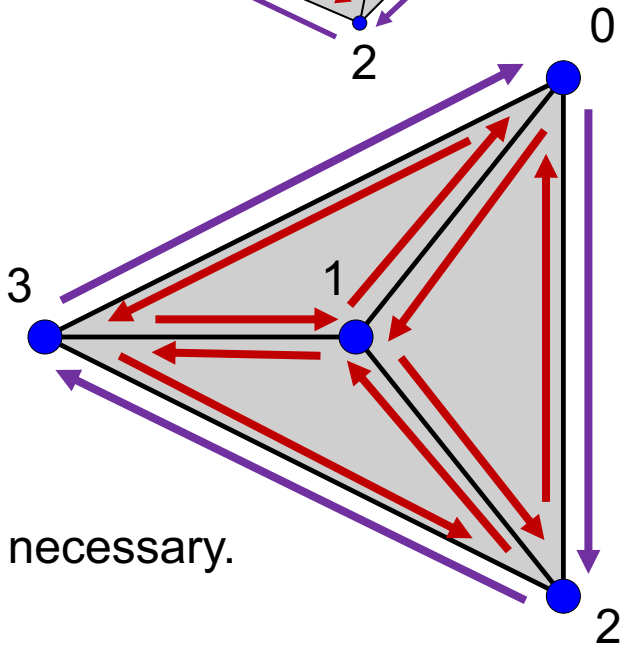
```
HalfEdge* any_half_edge
```

Polytope

```
FaceList flist
VertexList vlist
```

Half-Edge Structures – Clip & Cap



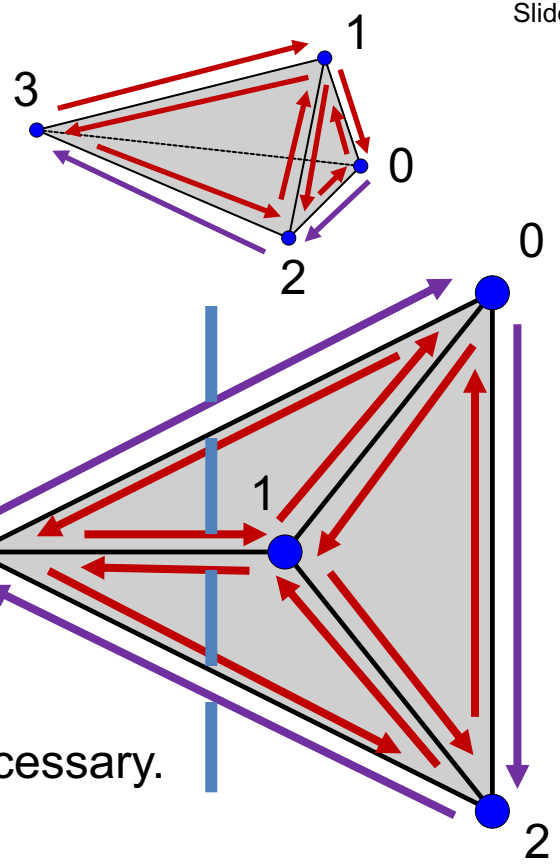


General algorithm

1. Calculate distance to vertices from plane
2. Find intersections and subdivide edges
3. Truncate each intersected face
4. Generate new faces on clip plane
5. Cleanup face and vertex list

Half edge structure provides all connectivity information necessary.
Allows for fast intersection of any polytope with a plane.

Half-Edge Structures – Clip & Cap

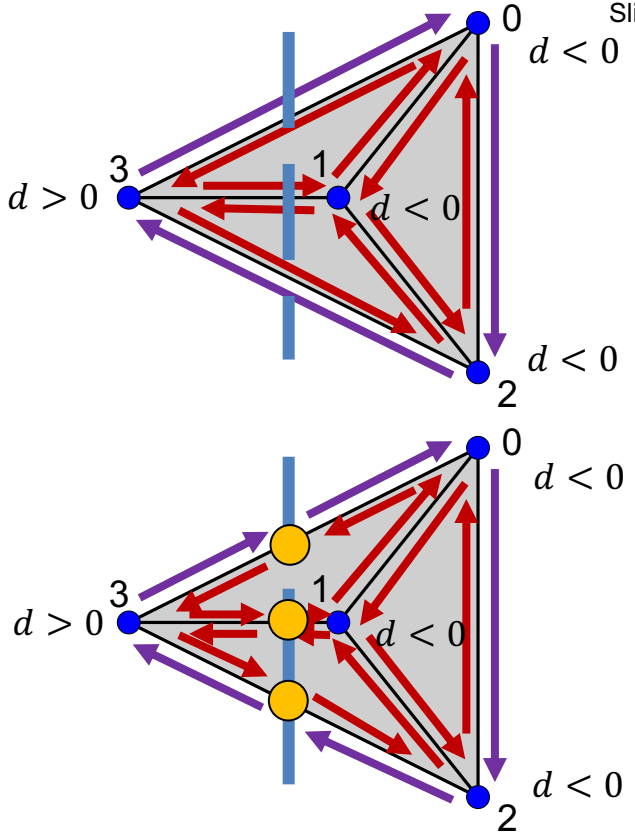


General algorithm

1. Calculate distance to vertices from plane
2. Find intersections and subdivide edges
3. Truncate each intersected face
4. Generate new faces on clip plane
5. Cleanup face and vertex list

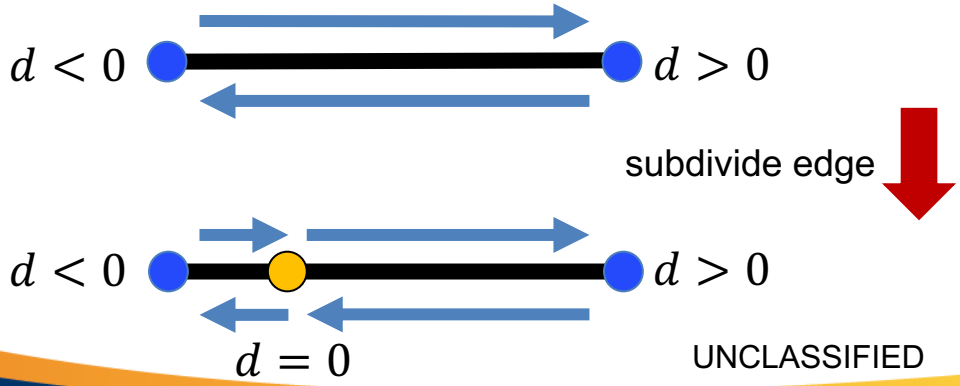
Half edge structure provides all connectivity information necessary.
Allows for fast intersection of any polytope with a plane.

Half-Edge Structures – Clip & Cap

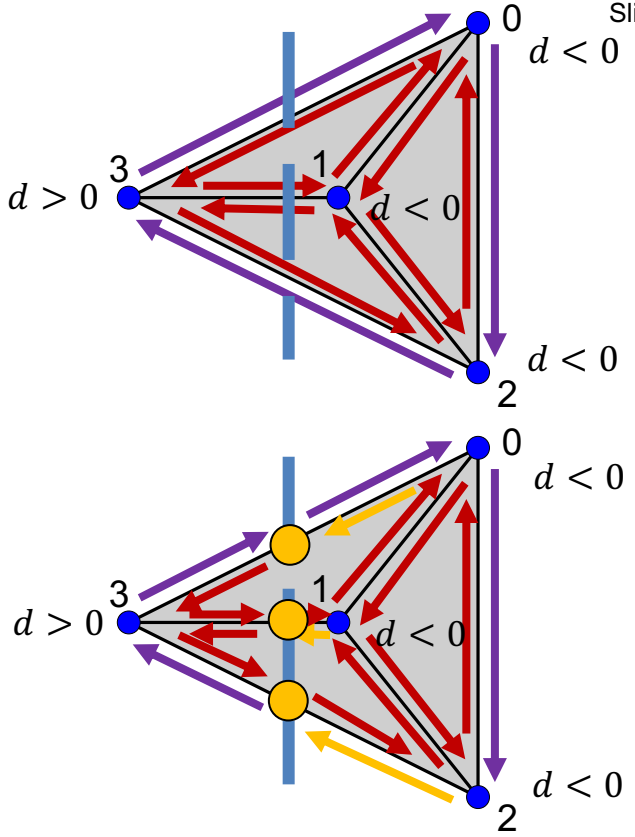


Find intersections and subdivide edges

1. Mark vertices with $d > 0$ to be clipped
2. Loop over all vertices marked for clipping
   a) Look at edges connected to neighboring vertices
   b) If intersection found, subdivide the edge
   c) Set vertex's half edge as new half edge

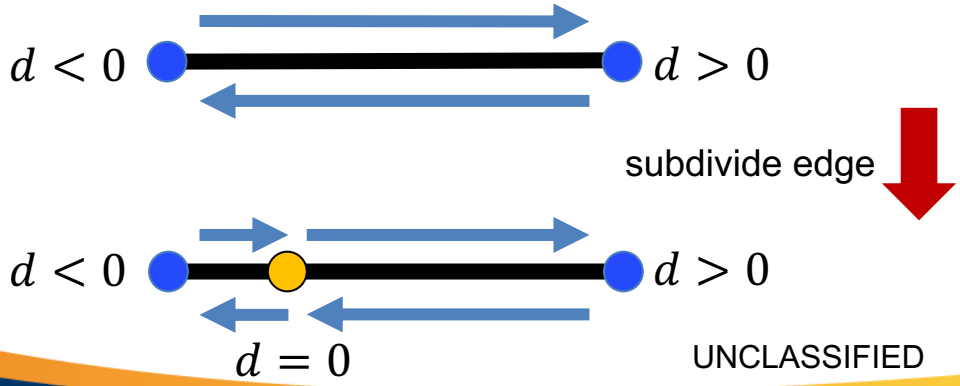


Half-Edge Structures – Clip & Cap

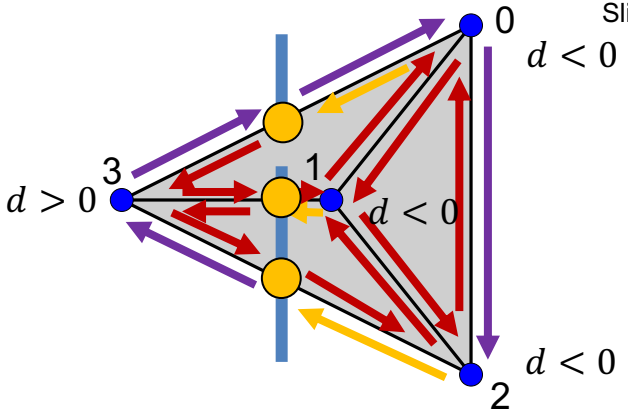


Find intersections and subdivide edges

1. Mark vertices with $d > 0$ to be clipped
2. Loop over all vertices marked for clipping
   a) Look at edges connected to neighboring vertices
   b) If intersection found, subdivide the edge
   c) Set vertex's half edge as new half edge



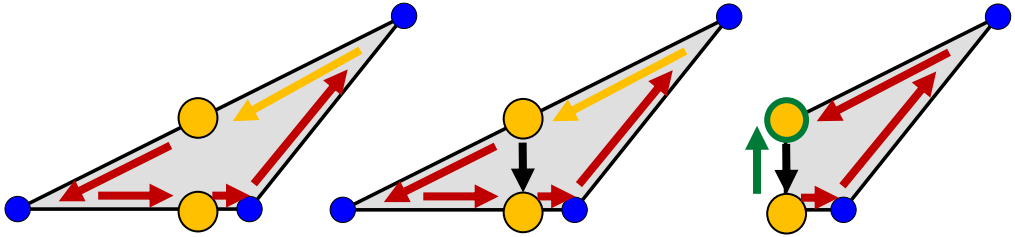
Half-Edge Structures – Clip & Cap

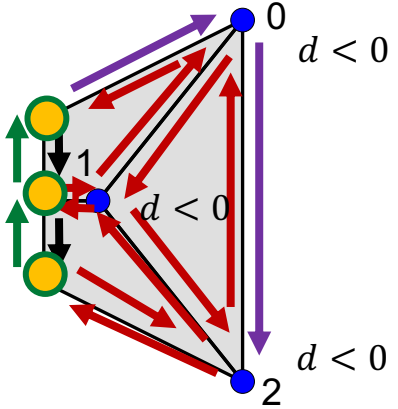


Truncate each intersected face

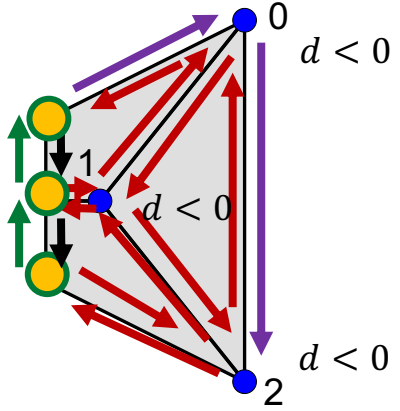
Loop over all new vertices

a) Traverse face until finding next new vertex
b) Make new half edge between two new vertices
c) Create a mirror half edge for new half edge
d) Set starting vertex's half edge as new mirror





Half-Edge Structures – Clip & Cap



Generate new faces on clip plane

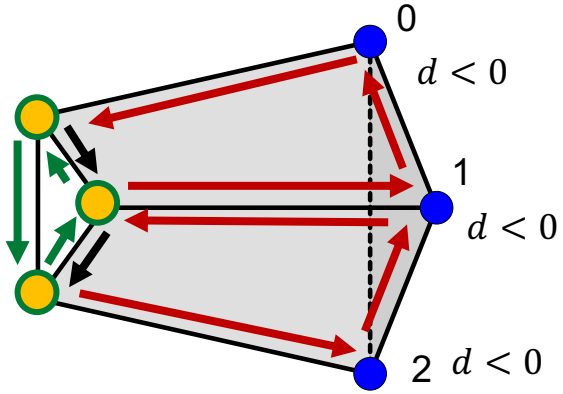
Loop over all new vertices

a) If vertex has not been visited, take half edge

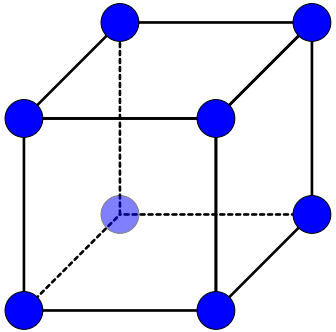
b) Create a new face and add to list

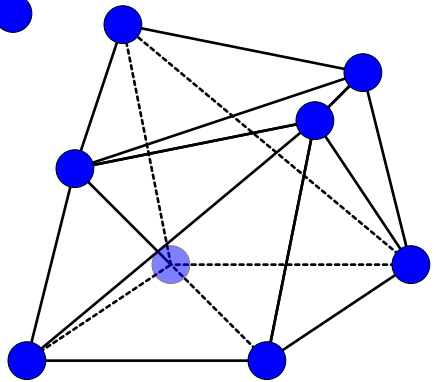
c) Link all half edges on face

```
current_half_edge = starting_half_edge
do
    half_edge.vertex.visited = true
    opposite_half_edge =
        current_half_edge.opposite
    previous_vertex = opposite_half_edge.vertex
    current_half_edge.previous =
        previous_vertex.half_edge
    previous_vertex.half_edge =
        current_half_edge
    current_half_edge =
        current_half_edge.previous
while (current_half_edge != starting_half_edge)
```



Clip & Cap Performance





Intersection of random planes with polyhedron

Generate m random planes, for $m \in [1,5]$

- random normal $\hat{n}$
- Distance in range [-0.1,0.1]

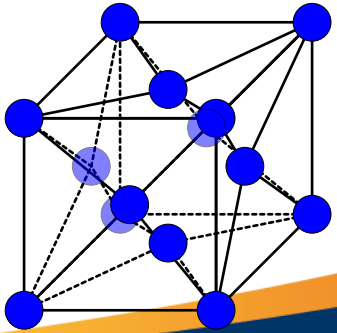
Polyhedrons intersected: Unit Cube, Dodecahedron, Poly24

All centered around the origin [0,0,0]

Intersection of two polyhedra

Two random tetrahedra with vertices on the unit sphere

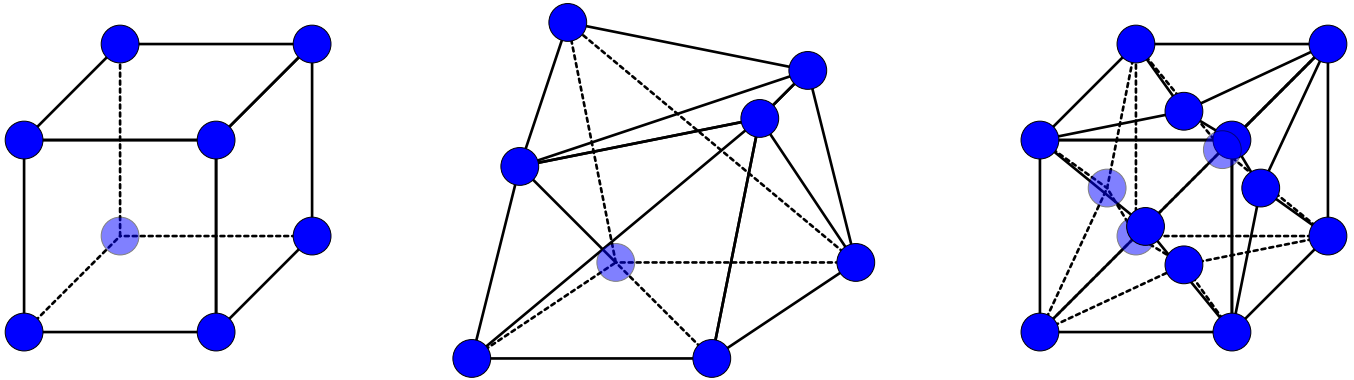
Two random axis-aligned cuboids with vertices on random spheres



Clip & Cap Performance

R3D

- C Library to perform intersection of polyhedra with planes
- Based on simplification of half-edge structure to a three-vertex connected graph
- Provides routines for voxelization of polyhedral to rectangular grid



Clip & Cap Performance

Timing in milliseconds / 10^6 trials

Unit Cube Intersections

m	IRL	R3D	IRL/ R3D
1	649	1320	0.49
2	1240	1890	0.66
3	1790	2300	0.78
4	2260	2620	0.86
5	2660	2840	0.94

Dodecahedron Intersections

m	IRL	R3D	IRL/ R3D
1	1050	2160	0.49
2	1970	2800	0.70
3	2500	3260	0.77
4	3070	3850	0.80
5	3520	3740	0.94

Poly24 Intersections

m	IRL	R3D	IRL/ R3D
1	1500	3860	0.39
2	3040	4410	0.69
3	3550	4550	0.78
4	4530	4590	0.99
5	5250	5370	0.98

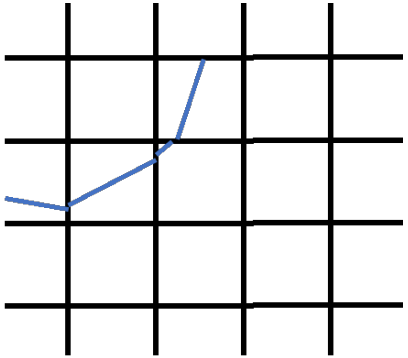
Two polyhedra intersections

Intersection Type	IRL	R3D	IRL/ R3D
Tet-Tet	2030	2170	0.94
Cuboid-Cuboid	1220	1400	0.87

Intersection of a Volume onto a Mesh

Relevant in many fields

- Geometric advection in multiphase flow simulations
- Conservative grid remapping schemes
- Voxelization in graphics



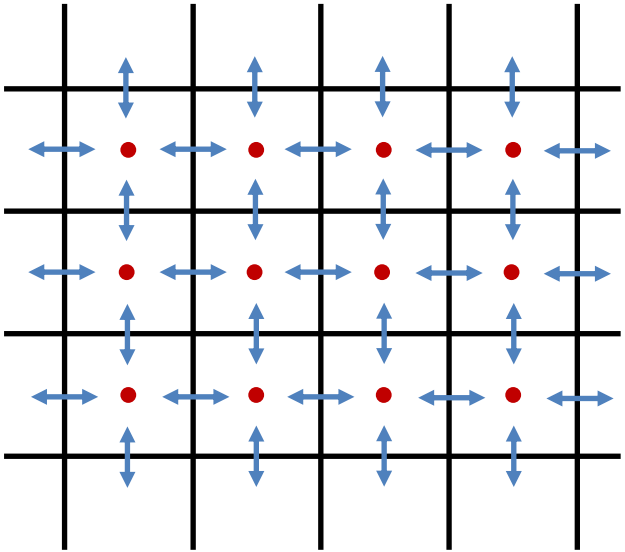
Geometric Advection

Implemented in IRL with planar reconstructions as nodes in a graph

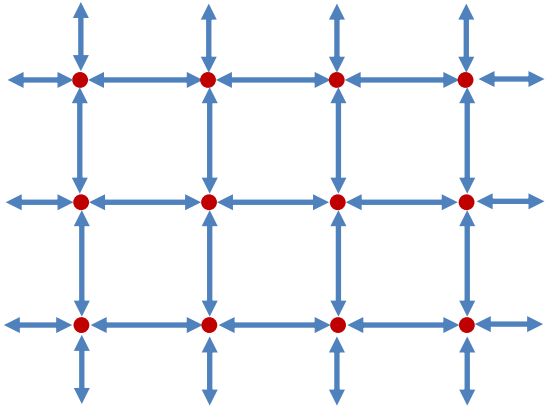
- Capable of representing any mesh of convex cells
- Includes information on volume attributed to each node

Intersection of a Volume onto a Mesh

Distribution of Volume onto a Mesh Using Reconstruction Links

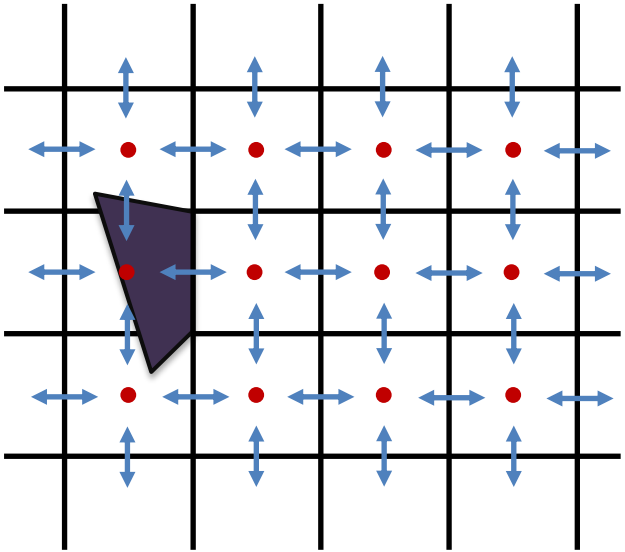


Bi-Directional Graph Connecting Reconstructions

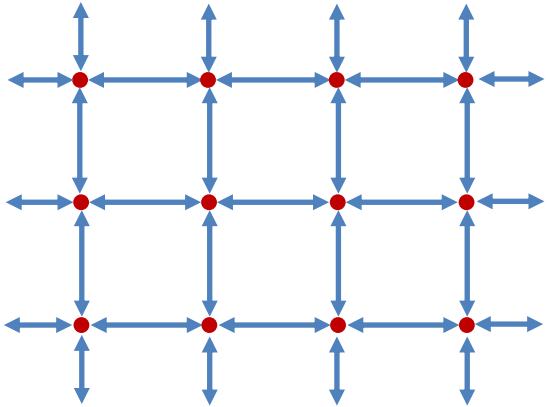


Intersection of a Volume onto a Mesh

Distribution of Volume onto a Mesh Using Reconstruction Links

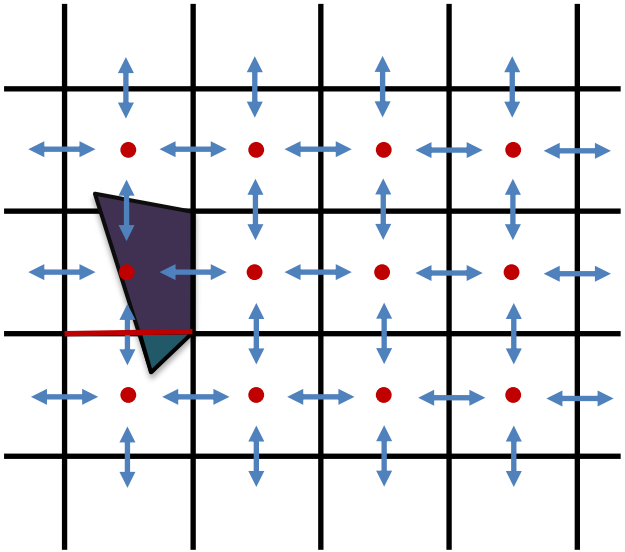


Bi-Directional Graph Connecting Reconstructions

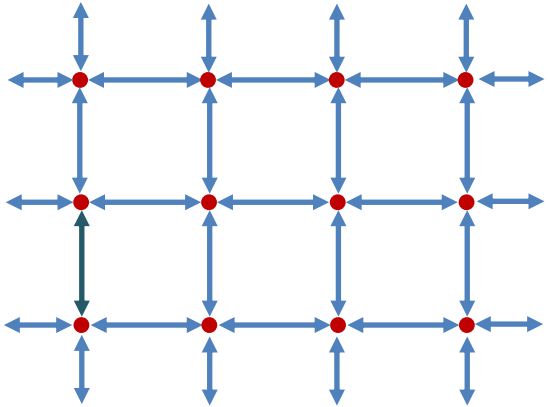


Intersection of a Volume onto a Mesh

Distribution of Volume onto a Mesh Using Reconstruction Links

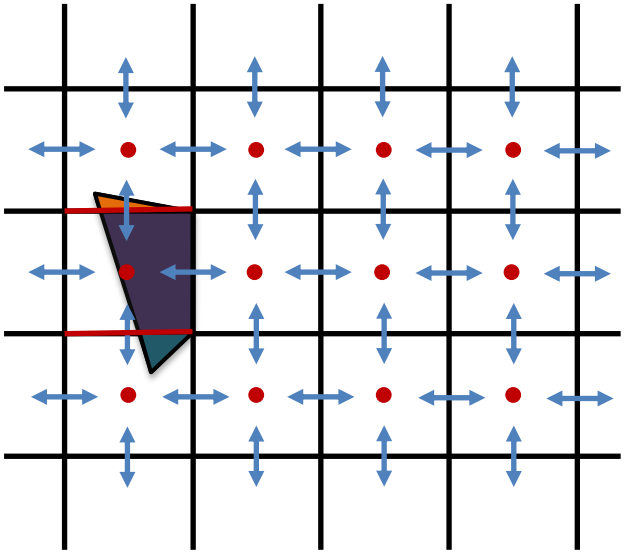


Bi-Directional Graph Connecting Reconstructions

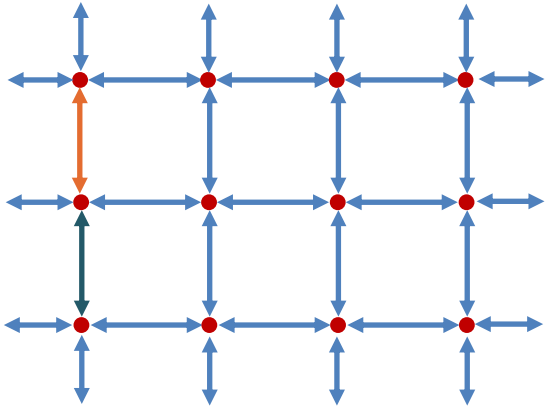


Intersection of a Volume onto a Mesh

Distribution of Volume onto a Mesh Using Reconstruction Links

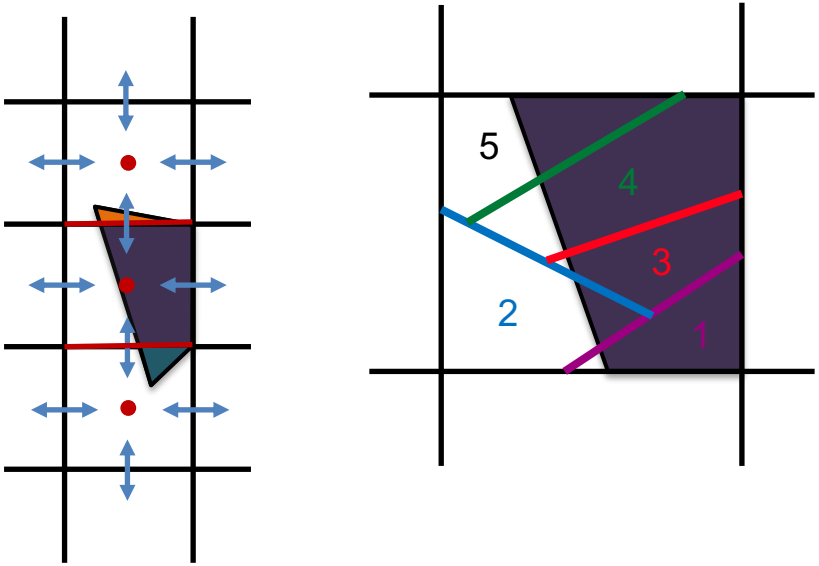


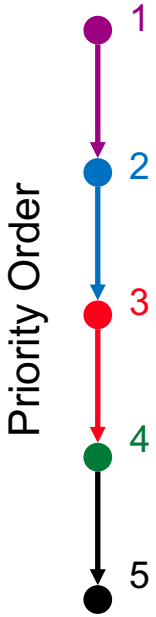
Bi-Directional Graph Connecting Reconstructions



Intersection of a Volume onto a Mesh

Dividing of Volume into Multiple Materials Using Reconstruction Links





Directed Path Graph Connecting Reconstructions

Performing nested dissection uses the same algorithms

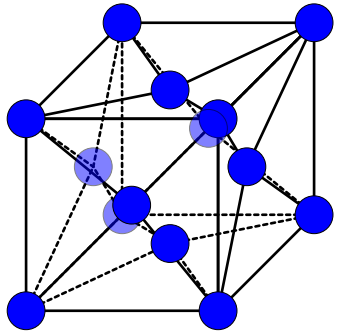
Voxelization Performance

Intersection of polyhedron onto a mesh

- Intersect with a cubic mesh in domain $[-1,1]^3$
- Mesh resolved by 3 cells in each direction
- For r3d, use `r3d_voxelize` routine

Polyhedra intersected:

- Random tetrahedron with vertices on unit sphere
- Random cuboid with vertices on unit sphere
- Concave Poly24 randomly translated in range [-0.5,0.5]

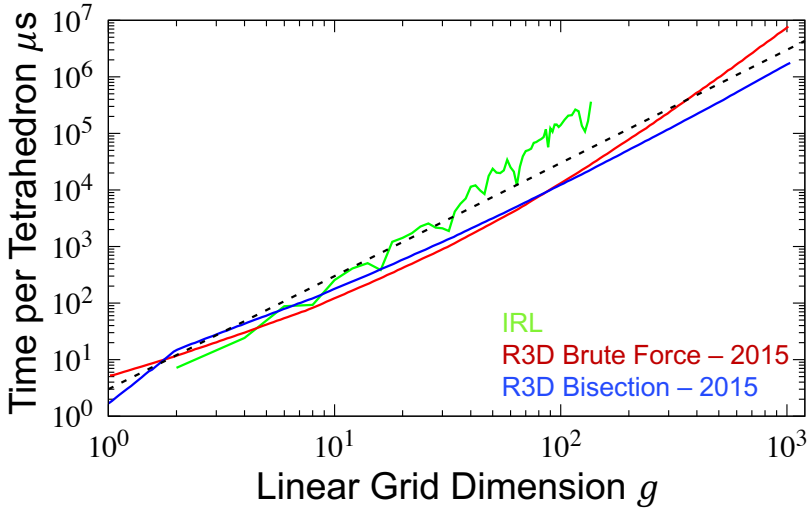


Voxelization Performance

Intersection of polyhedron onto a mesh

Intersection Type	IRL	R3D	IRL/ R3D
Tetrahedron	13670	30560	0.45
Cuboid	9550	16550	0.58
Poly24	8920	67070	0.13

Timing in milliseconds / 10^6 trials



**IRL averaged over 10^4 trials*

Interface Reconstruction Library (IRL)

IRL provides a robust, efficient, and general volume intersection algorithm

- Works for polyhedra and polygons
- No inherent limit on number of vertices, faces, or half edges
- Planes can be linked together to form a graph for efficient local mesh intersection

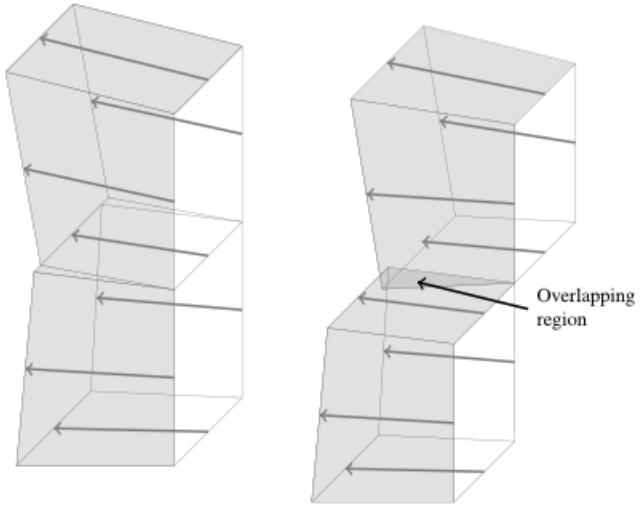
Other features of IRL

- Contains interface reconstruction methods: LVIRA, MoF, R2P, Advected Normals
- Volume conservation enforcement for planar reconstructions
- General root finding and optimization framework

Efficient Conservative Unsplit Advection for Unstructured Meshes

Construction of Consistent Fluxes for Structured Meshes

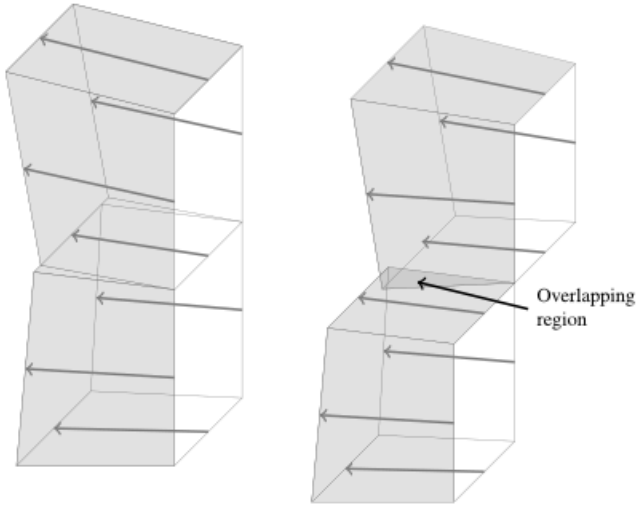
Using edge velocity
(overlapping and gaps)



Owkes & Desjardins. JCP, 2014

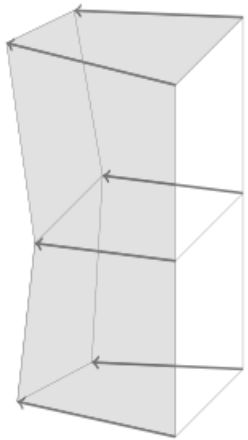
Construction of Consistent Fluxes for Structured Meshes

Using edge velocity
(overlapping and gaps)



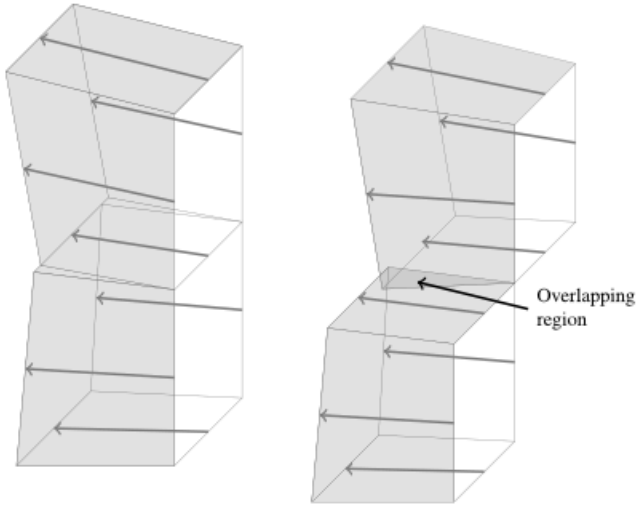
Owkes & Desjardins. JCP, 2014

Using node velocity
(fixed overlapping/gaps)



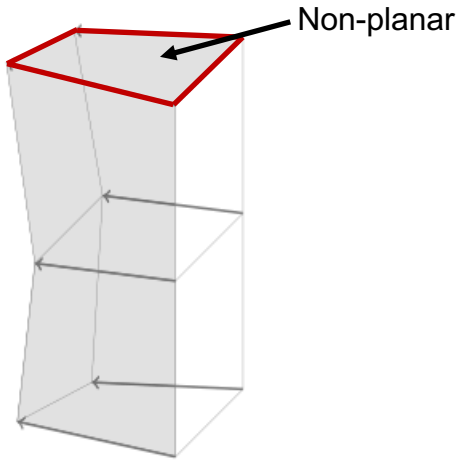
Construction of Consistent Fluxes for Structured Meshes

Using edge velocity
(overlapping and gaps)



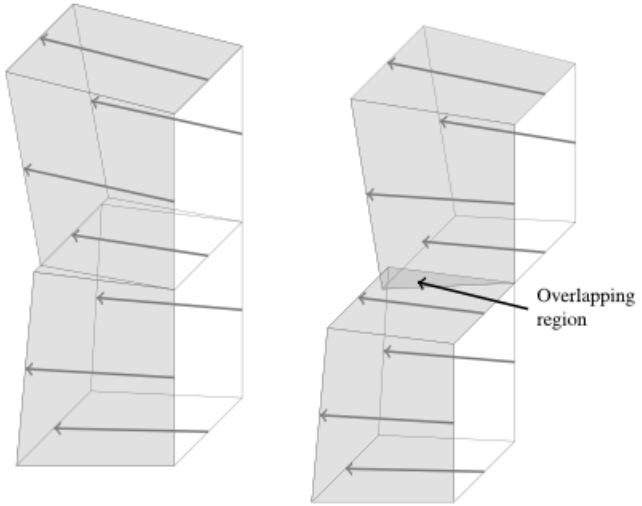
Owkes & Desjardins. JCP, 2014

Using node velocity
(fixed overlapping/gaps)
(non-planar faces)



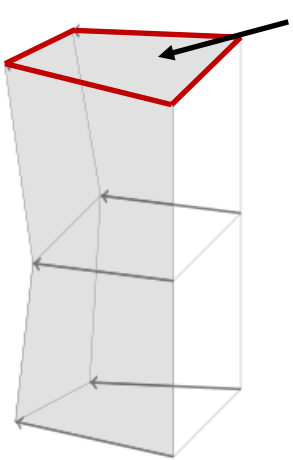
Construction of Consistent Fluxes for Structured Meshes

Using edge velocity
(overlapping and gaps)

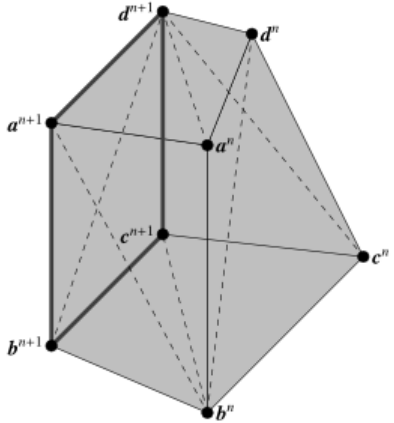


Owkes & Desjardins. JCP, 2014

Using node velocity
(fixed overlapping/gaps)
(non-planar faces)



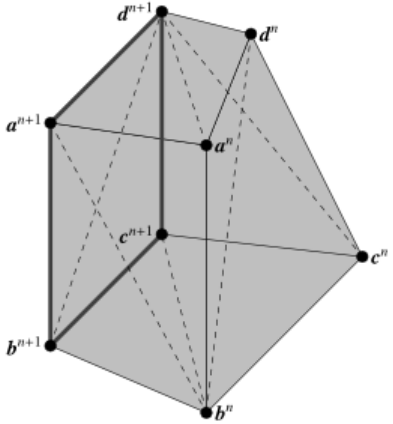
Triangulate face to force coplanarity



Owkes & Desjardins. JCP, 2014

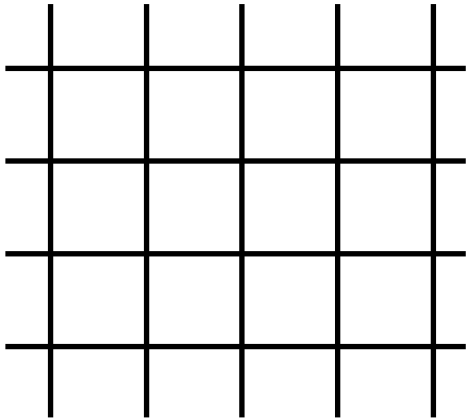
Construction of Consistent Fluxes for Structured Meshes

Imposed triangulation

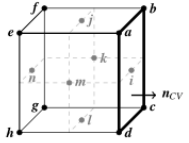


Owkes & Desjardins. JCP, 2014

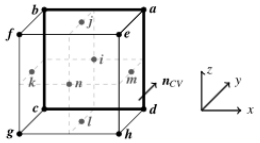
Structured mesh



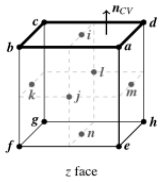
Pre-tabulated vertex orderings



6 Simplices				
1	g	d	c	f
2	f	c	b	d
3	f	b	a	d
4	g	h	d	f
5	f	e	h	d
6	f	e	d	a

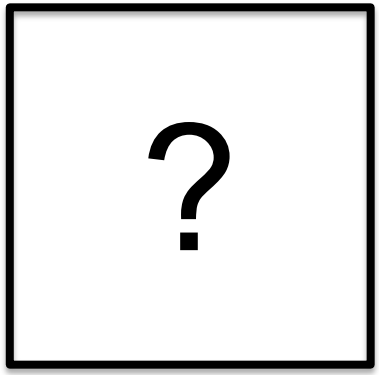


20 Simplices				
1	a	e	j	i
2	e	f	j	i
3	f	b	j	i
4	b	a	j	i
5	b	f	k	i
6	f	g	k	i
7	g	c	k	i
8	c	b	k	i
9	c	g	l	i
10	g	h	l	i
11	h	d	l	i
12	d	c	l	i
13	d	h	m	i
14	h	e	m	i
15	e	a	m	i
16	a	d	m	i
17	e	h	n	i
18	h	g	n	i
19	g	f	n	i
20	f	e	n	i

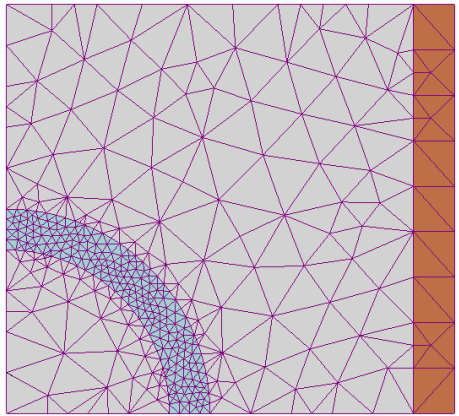


Construction of Consistent Fluxes for Unstructured Meshes

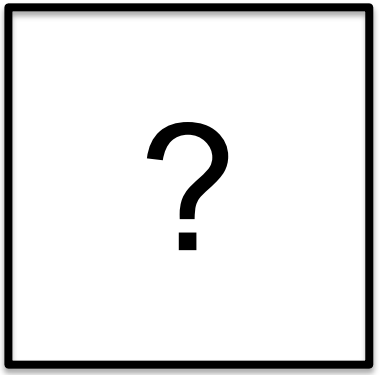
Imposed triangulation



Unstructured mesh



wikipedia.org



Construction of Consistent Fluxes for Unstructured Meshes

Truchas has four types of elements

1. Tetrahedron
2. Pyramid
3. Wedge (Triangular Prism)
4. Hexahedron

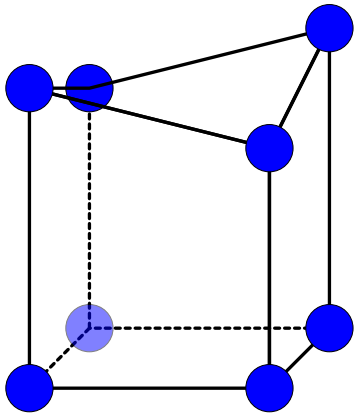
All of these have faces with 3 or 4 vertices

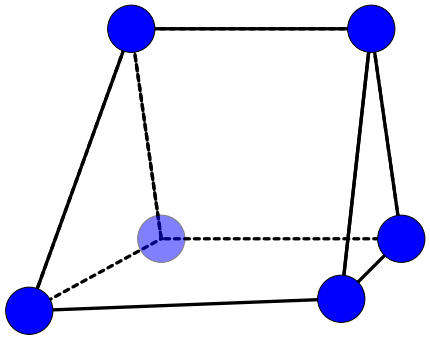
→ flux volumes with 6 or 10 vertices

In general, the faces of the flux volumes are **NOT** coplanar

Construction of Consistent Fluxes for Unstructured Meshes

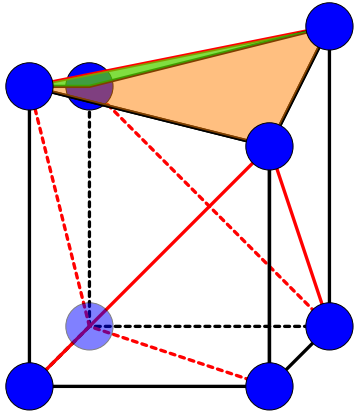
Invalid Hexahedron

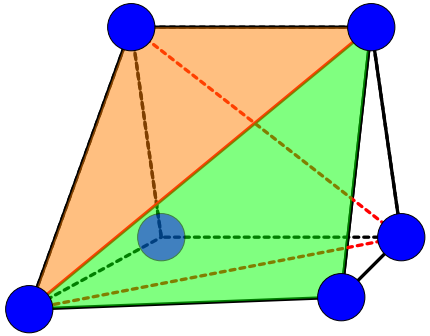




Invalid Wedge

Valid Dodecahedron



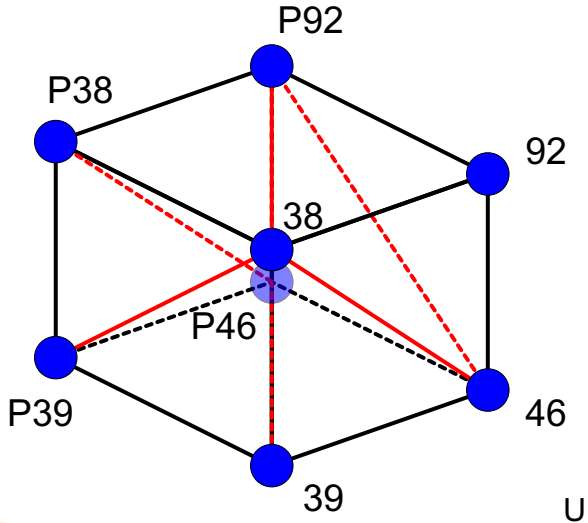


Valid Octahedron

Construction of Consistent Fluxes for Unstructured Meshes

1. Triangulation is necessary to force coplanarity
2. Triangulation itself is arbitrary

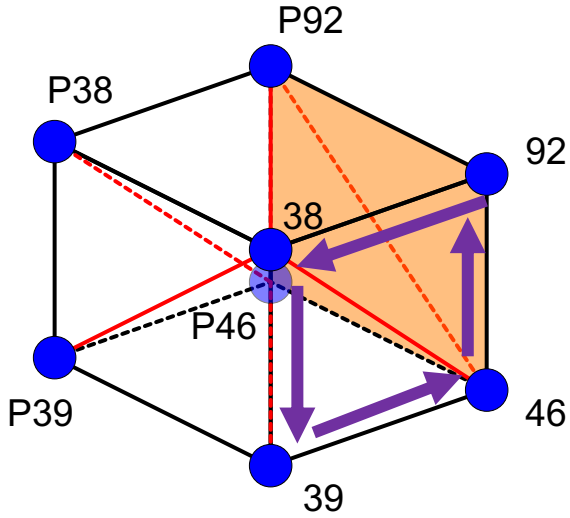
Choice: Diagonalize projected edge faces based on ascending global node id

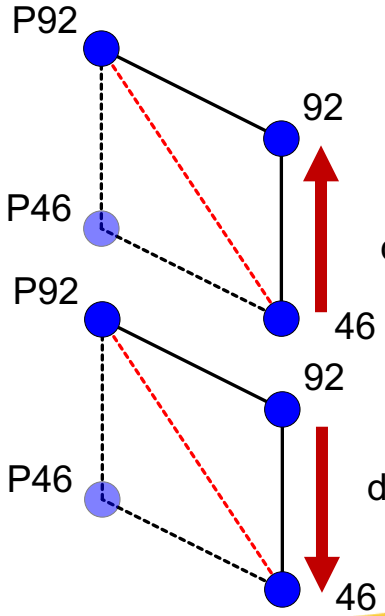


Construction of Consistent Fluxes for Unstructured Meshes

Flux volumes must be constructed from known face triangulations

→ Codify the face diagonalization relative to the traversal of the face





"Leading" (L)
diagonalization follows traversal

"Trailing" (T)
diagonalization opposite traversal

Construction of Consistent Fluxes for Unstructured Meshes

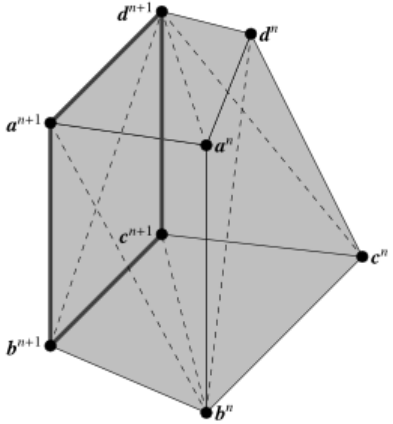
Ten categories of flux polyhedra result from this codification

Octahedron
LLL
LLT
LTT
TTT

Dodecahedron
LLLL
LLLT
LTLT
LLTT
LTTT
TTTT

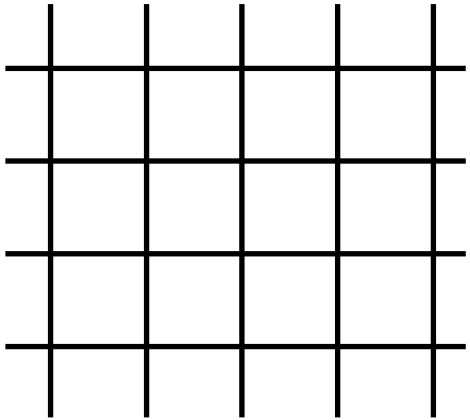
Construction of Consistent Fluxes for Structured Meshes

Imposed triangulation

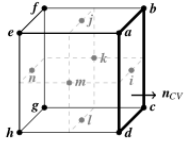


Owkes & Desjardins. JCP, 2014

Structured mesh

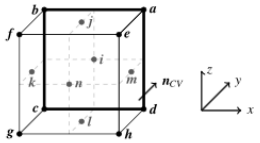


Pre-tabulated vertex orderings



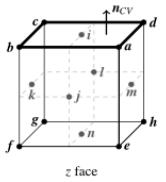
6 Simplices

1	g	d	c	f
2	f	c	b	d
3	f	b	a	d
4	g	h	d	f
5	f	e	h	d
6	f	e	d	a



20 Simplices

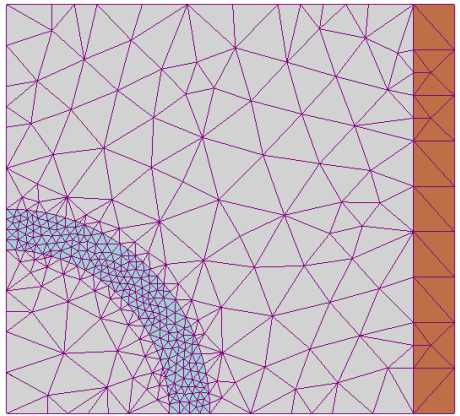
1	a	e	j	i
2	e	f	j	i
3	f	b	j	i
4	b	a	j	i
5	b	f	k	i
6	f	g	k	i
7	g	c	k	i
8	c	b	k	i
9	c	g	l	i
10	g	h	l	i
11	h	d	l	i
12	d	c	l	i
13	d	h	m	i
14	h	e	m	i
15	e	a	m	i
16	a	d	m	i
17	e	h	n	i
18	h	g	n	i
19	g	f	n	i
20	f	e	n	i



Construction of Consistent Fluxes for Unstructured Meshes

Imposed triangulation

Octahedron	Dodecahedron
LLL	LLLL
LLT	LLLT
LTT	LTLT
TTT	LLTT
	LTTT
	TTTT



wikipedia.org

Unstructured mesh

Vertex Ordering Set at Initialization

ordered_nodes

polyhedra_id

Construction of Consistent Fluxes for Unstructured Meshes

Classification of flux volumes types only depends on mesh elements and connectivity

- Can be performed during initialization, limited impact on performance
- General idea applicable for achieving conservative fluxing with any element geometry
- Cases can be further reduced if allow for reversal of face traversal (Reduces to six cases)

Matching of flux volume to a given amount handled by pyramidal correction

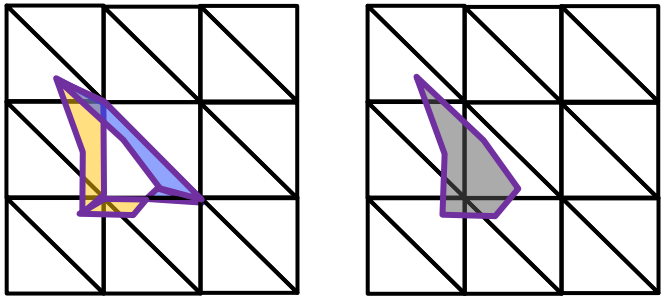
- Back projected face triangulated to a central point to find a “cap”
- Central point then adjusted to achieve desire volume
- Many options on deciding the line the central point is adjusted along

1. Owkes & Desjardins, JCP. 2014
2. Comminal et. al, JCP. 2015
3. Maríc et. al, JCP. 2018
4. Jofre et. al, C&F. 2014

Optimization of Unsplit Geometric VOF Advection

Cell-wise advection of volume fraction[1,2]

- Direct formation of final volume entering cell
- Reduces number of intersections on mesh
- Creates larger, better behaved polyhedra



Refinement of geometric advection band[3]

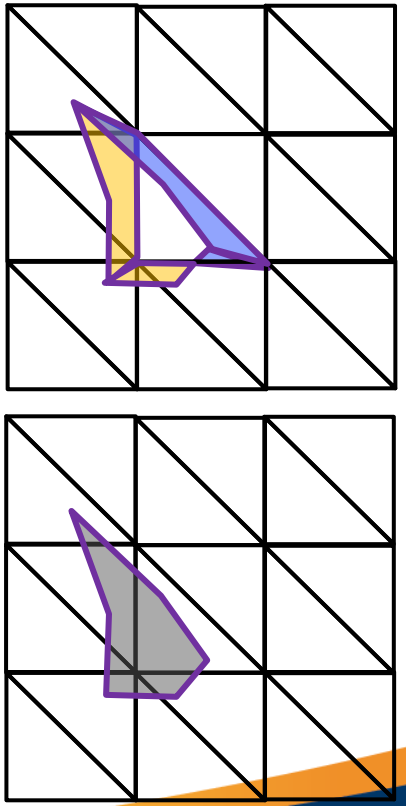
- Major cost is intersections with the mesh
- Approximates intersections with axis-aligned bounding boxes
- Fast and efficiently determines if intersection needed

1. Comminal et. al, JCP. 2015
2. Zhang, SIAM Num. Analysis. 2013
3. Maríc et. al, JCP. 2018

Optimization of Unsplit Geometric VOF Advection

Cell-wise advection of volume fraction

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha \vec{u}) = 0 \quad \begin{cases} \alpha^{n+1} = \alpha^{n} + \frac{1}{V_c} \sum_{f} F_{\alpha} \\ \alpha^{n+1} = \frac{1}{V_c} \left(P_{\Delta t}(C) \cap R_{\alpha} \right) \end{cases}$$



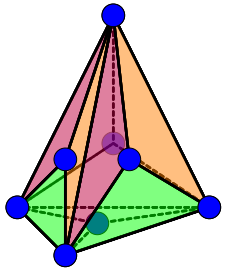
Algorithm

1. Back project cell vertices
2. Construct face fluxes (with correct face triangulation)
3. Correct face flux polyhedron to match given flux volume
4. Form corrected advected cell using projected vertices and correction vertices
5. Intersect corrected advected cell with mesh

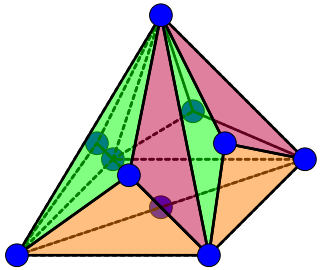
1. Comminal et. al, JCP. 2015
2. Zhang, SIAM Num. Analysis. 2013

Optimization of Unsplit Geometric VOF Advection

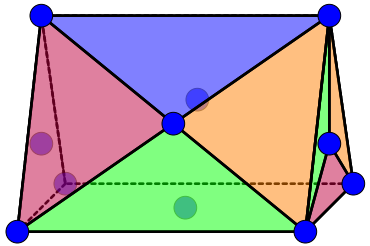
Cell-wise advection of volume fraction



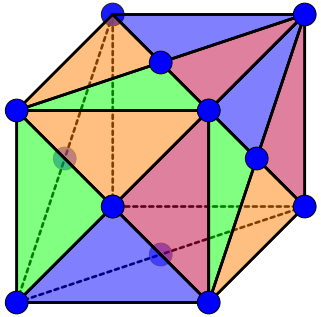
FP Tetrahedra



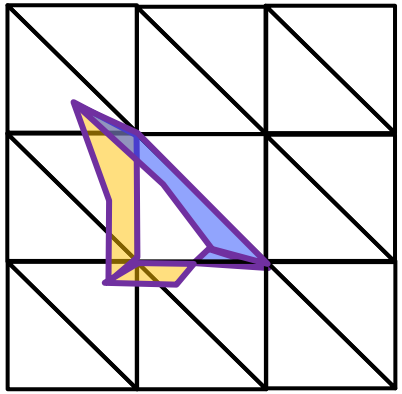
FP Pyramid

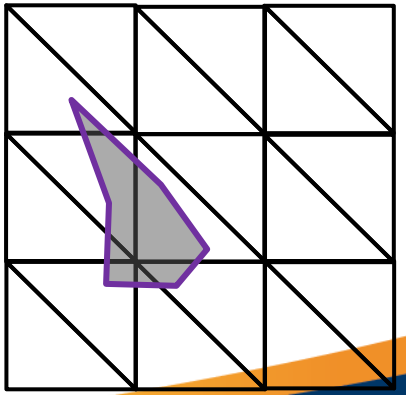


FP Wedge



FP Hexahedron

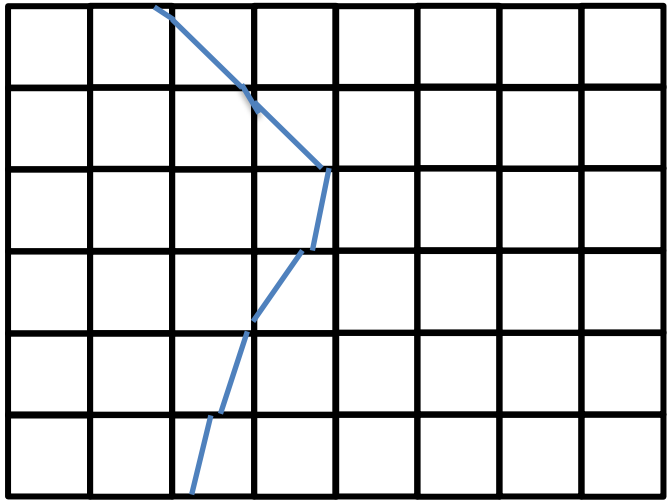




1. Maríc et. al, JCP. 2018

Optimization of Unsplit Geometric VOF Advection

Refinement of geometric advection band

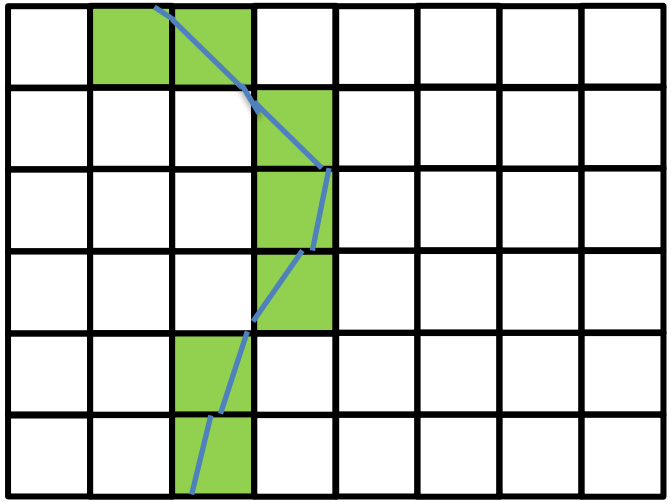


1. Maríc et. al, JCP. 2018

Optimization of Unsplit Geometric VOF Advection

Refinement of geometric advection band

Band 0

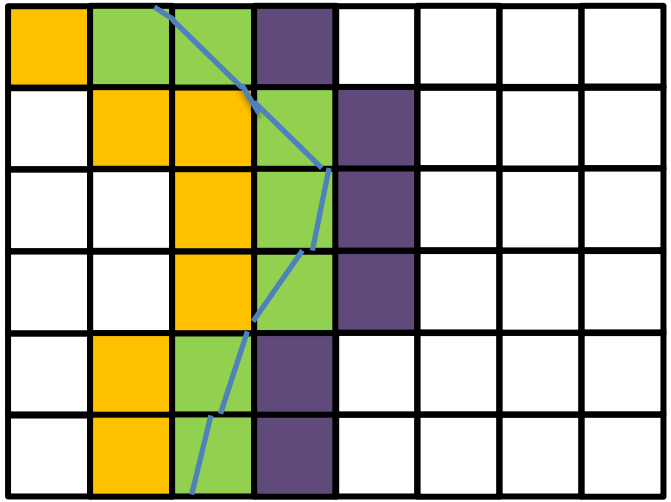


1. Maríc et. al, JCP. 2018

Optimization of Unsplit Geometric VOF Advection

Refinement of geometric advection band

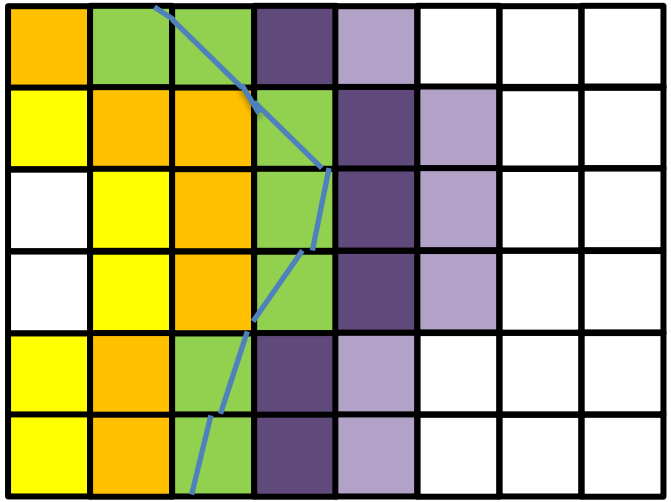
Band 0
Band +1
Band -1



Optimization of Unsplit Geometric VOF Advection

Refinement of geometric advection band

Band 0
Band +1
Band -1
Band +2
Band +2

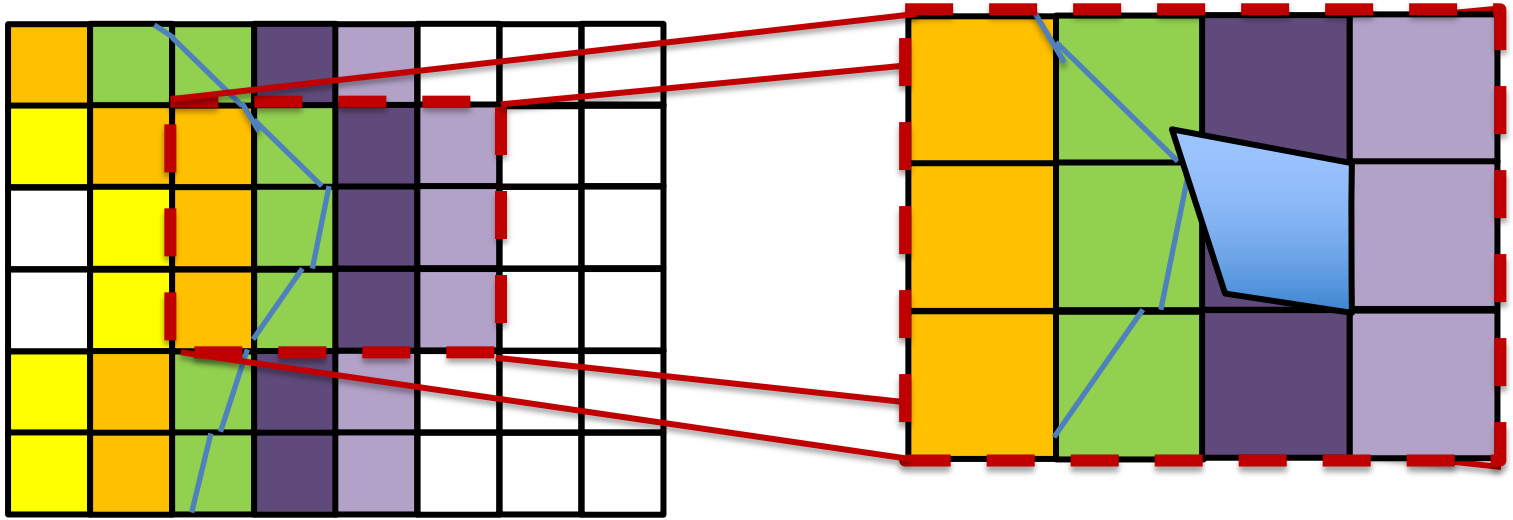


1. Maríc et. al, JCP. 2018

Optimization of Unsplit Geometric VOF Advection

Refinement of geometric advection band

Band 0
Band +1
Band -1
Band +2
Band +2

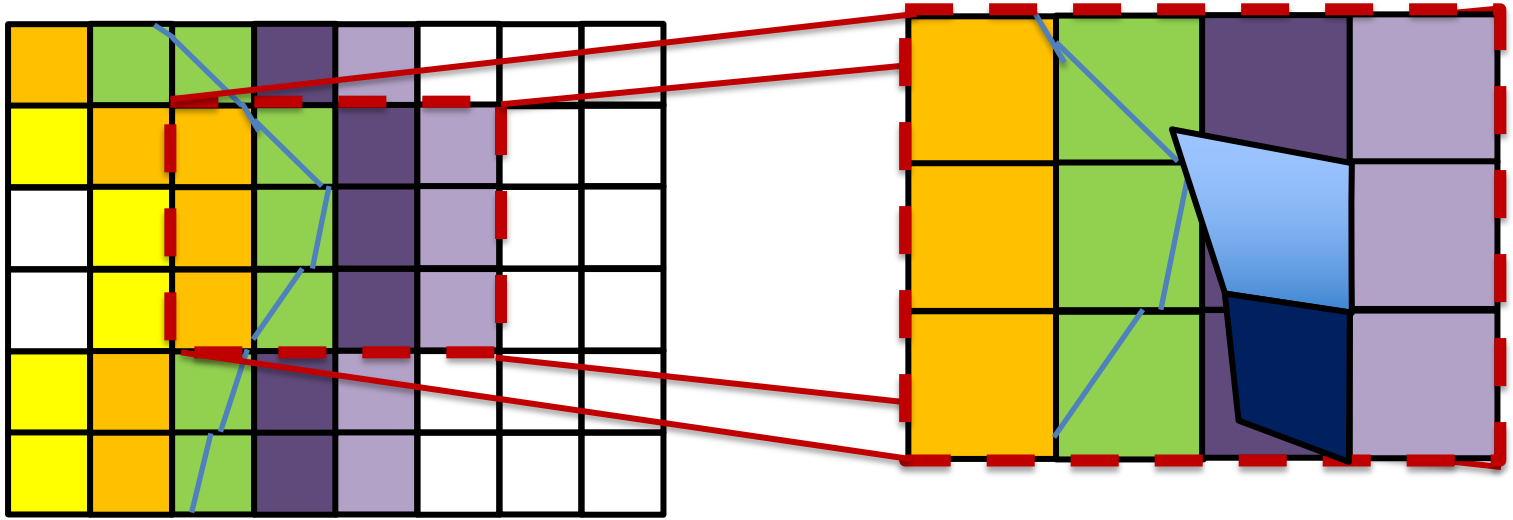


1. Maríc et. al, JCP. 2018

Optimization of Unsplit Geometric VOF Advection

Refinement of geometric advection band

Band 0
Band +1
Band -1
Band +2
Band +2

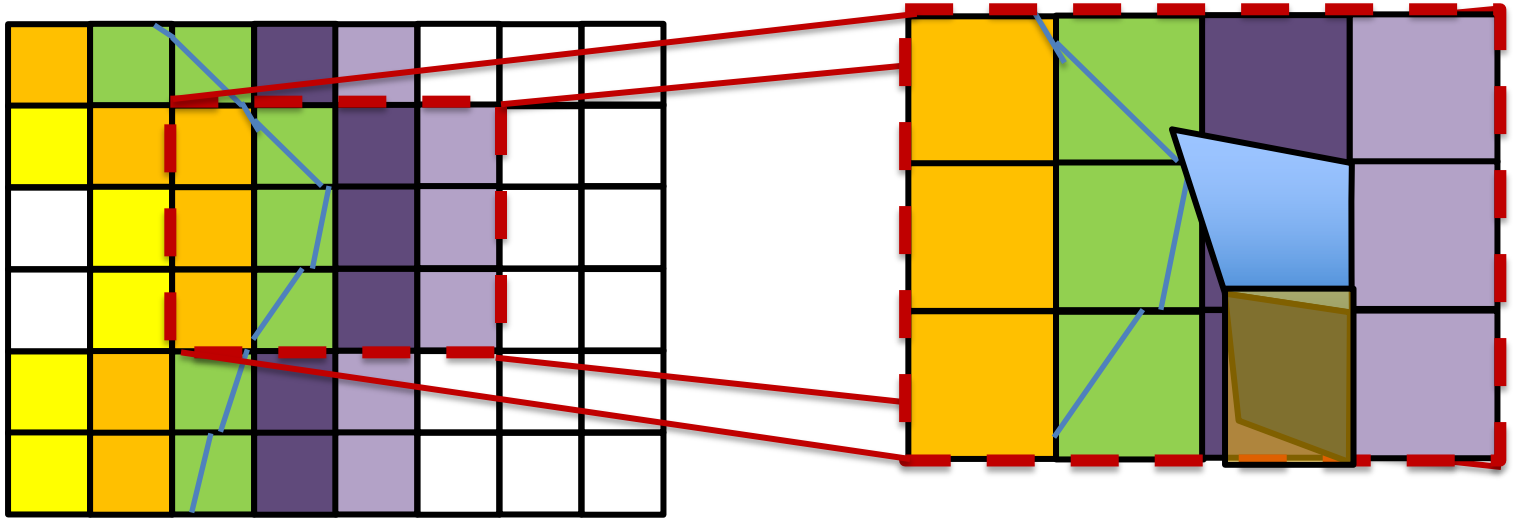


1. Maríc et. al, JCP. 2018

Optimization of Unsplit Geometric VOF Advection

Refinement of geometric advection band

Band 0
Band +1
Band -1
Band +2
Band +2



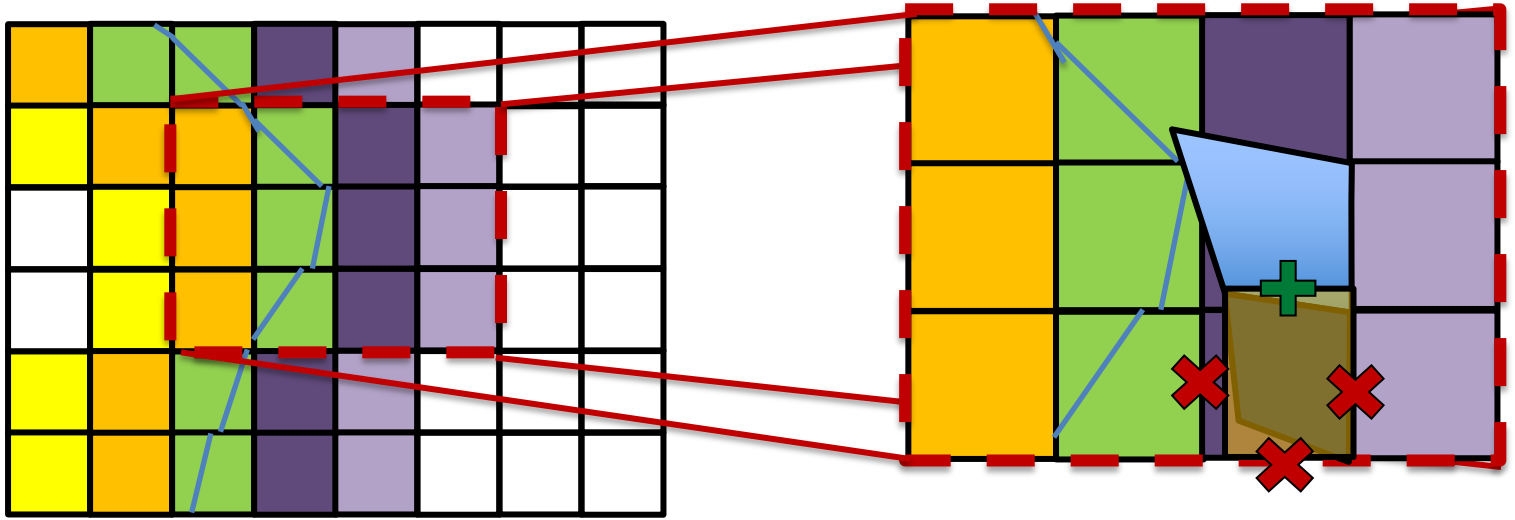
Does sign of band change?

1. Maríc et. al, JCP. 2018

Optimization of Unsplit Geometric VOF Advection

Refinement of geometric advection band

Band 0
Band +1
Band -1
Band +2
Band +2



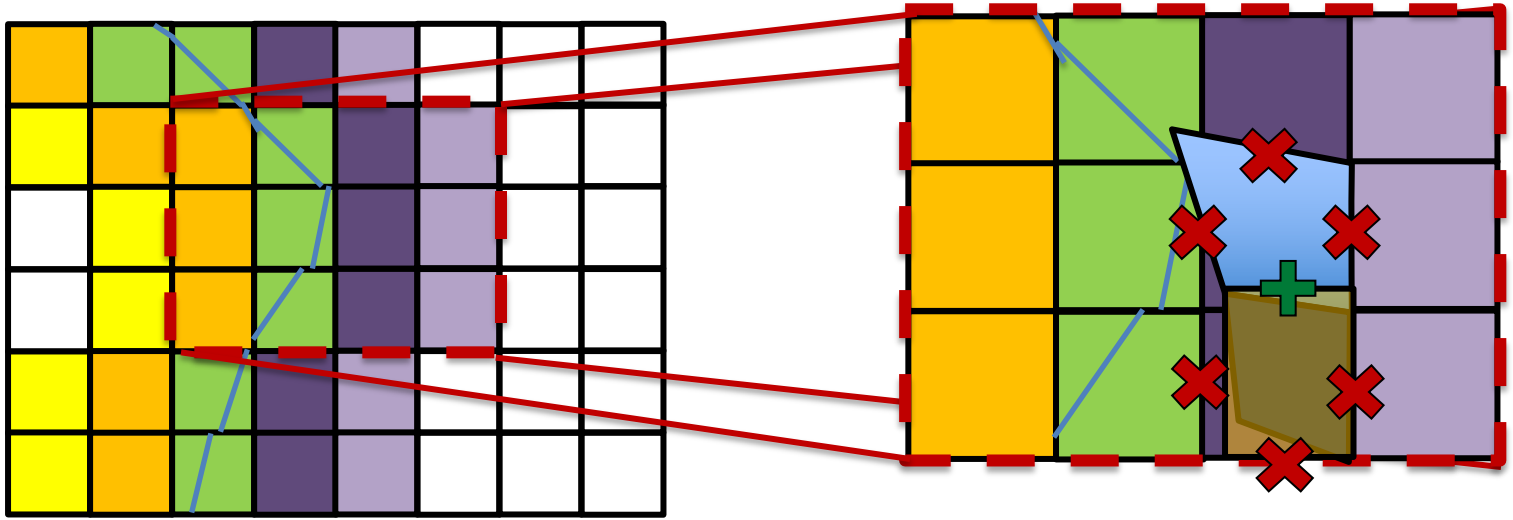
Does sign of band change?

1. Maríc et. al, JCP. 2018

Optimization of Unsplit Geometric VOF Advection

Refinement of geometric advection band

Band 0
Band +1
Band -1
Band +2
Band +2



Does sign of band change?

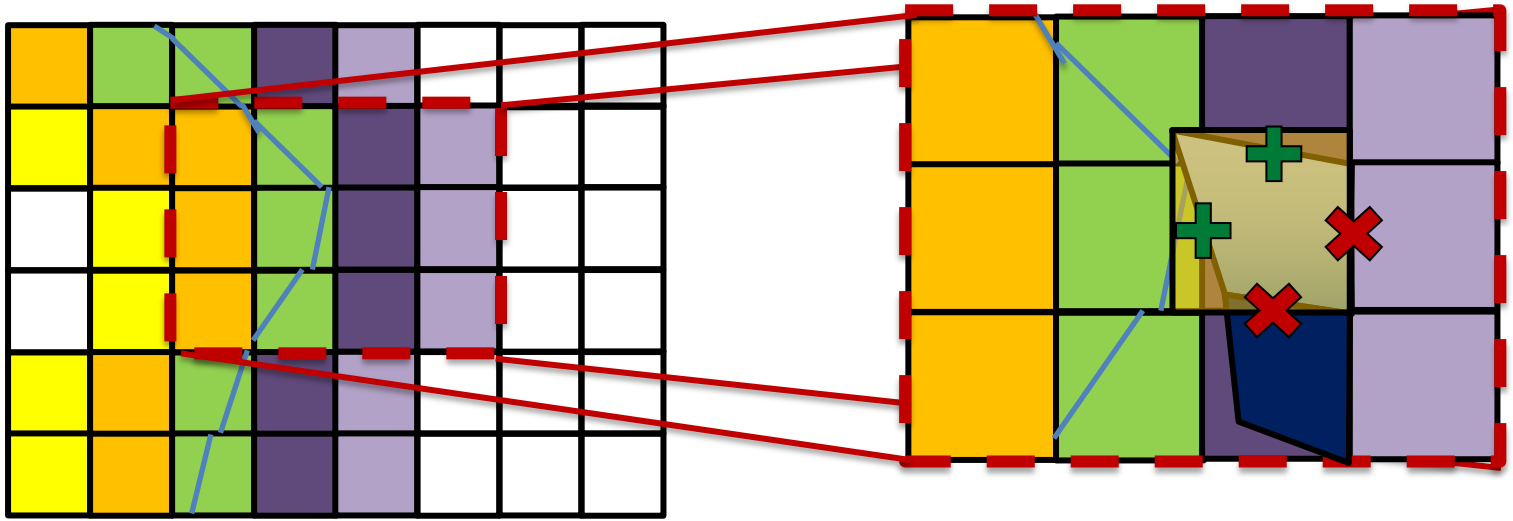
NO → single-phase flux

1. Maríc et. al, JCP. 2018

Optimization of Unsplit Geometric VOF Advection

Refinement of geometric advection band

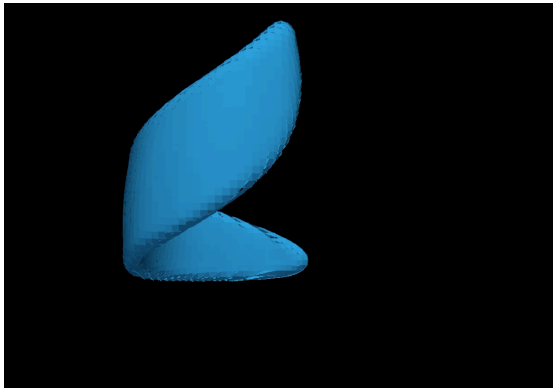
Band 0
Band +1
Band -1
Band +2
Band +2



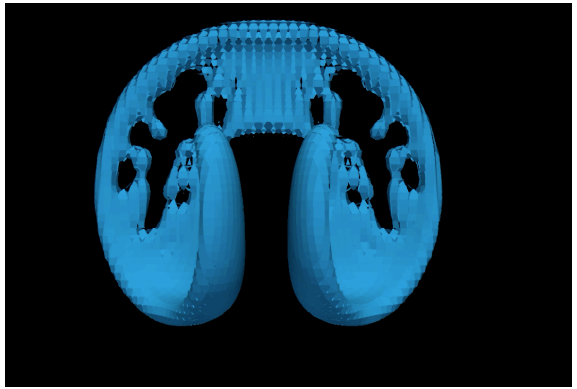
Does sign of band change?
YES ➡ multi-phase flux

Results – Two-Phase Advection in Truchas

3D Shear Case



3D Deformation Case



Performance measures

$$E_g = \frac{\sum(\alpha_{initial} - \alpha_{final})V}{\sum V}$$

$$E_v = \sum \alpha_{initial} V - \sum \alpha_{final} V$$

$$T_{cost} = \frac{T_{wall}}{n_{steps}}$$

Results – Two-Phase Advection in Truchas

3D Shear Case

Prescribed reversing velocity field

$$u(\vec{x},t) = \sin(2\pi y)\sin^2(\pi x)\cos(\pi t/3)$$
$$v(\vec{x},t) = -\sin(2\pi x)\sin^2(\pi y)\cos(\pi t/3)$$
$$w(\vec{x},t) = \left(1-\frac{r}{R}\right)^2\cos(\pi t/3),\ \ r = \sqrt{(x-0.5)^2+(y-0.5)^2},\ \ R = 0.5$$

<u>Domain</u>

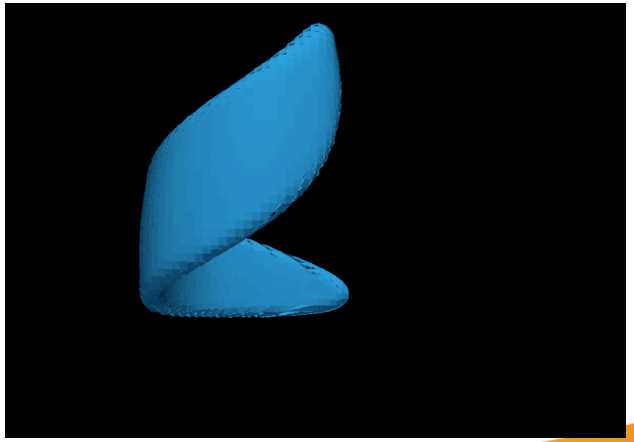
Dimensions $[0,0,0]\times[1,1,2]$

Discretized uniformly by $N\times N\times 2N$ hexahedron cells

Simulations performed at CFL of 0.5

<u>Initial Shape</u>

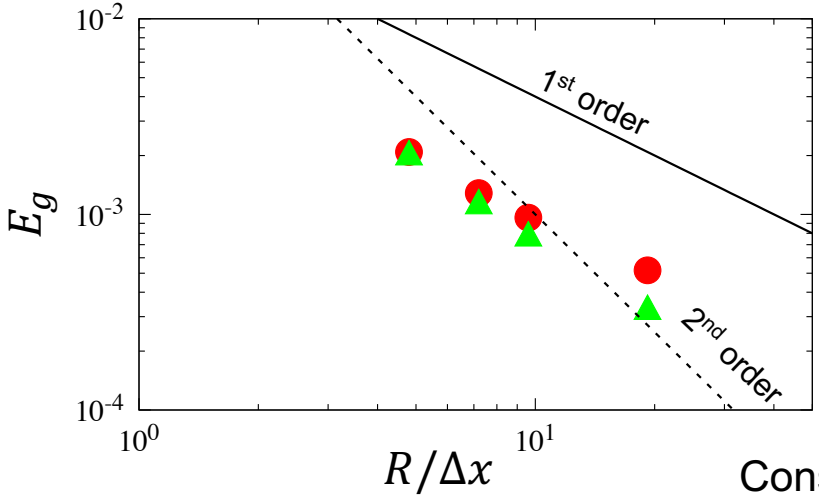
Sphere of radius 0.15, centered at $(0.5, 0.75, 0.25)$

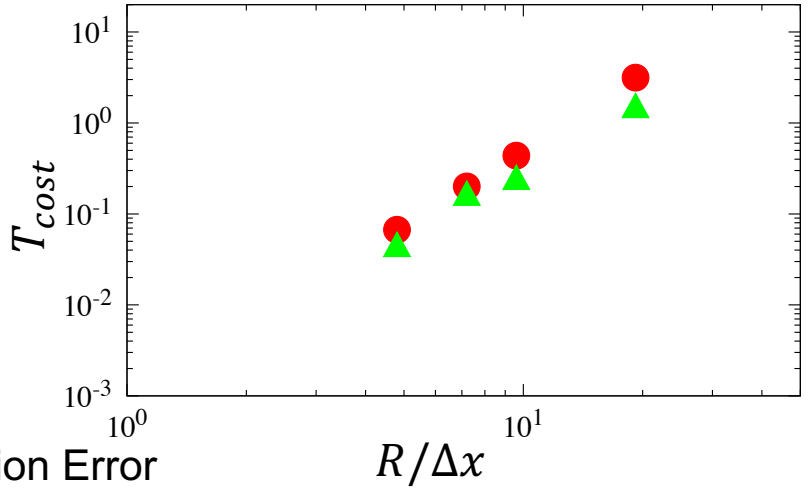


1. Liovic et. al, C&F. 2006

Results – Two-Phase Advection in Truchas

3D Shear Case





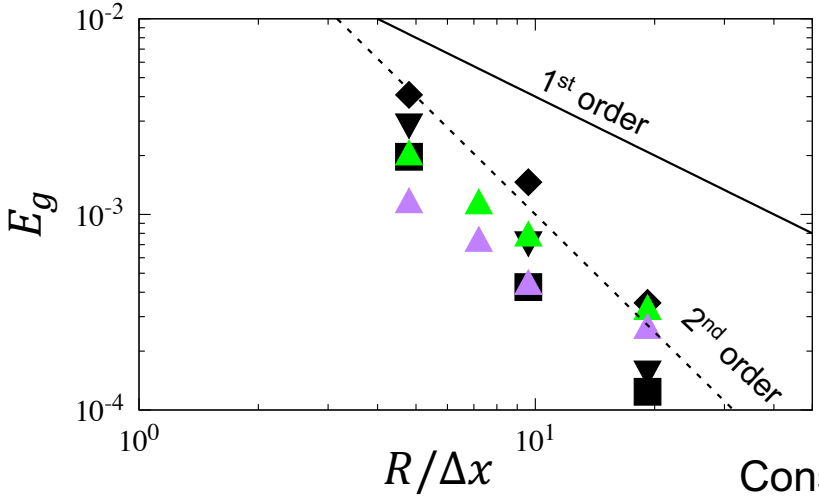
Conservation Error

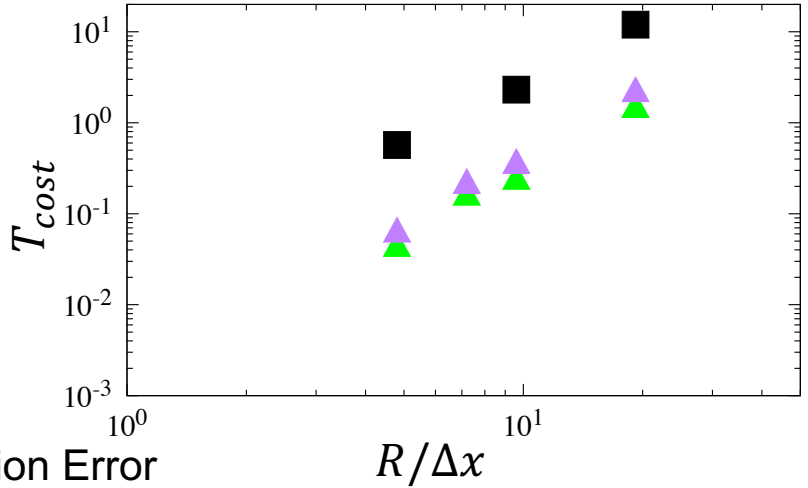
- Old Method (Youngs)
- New Transport (Youngs)

Resolution	New	Old
4.8	+1.076E-16	-1.943E-16
7.2	-3.530E-15	+5.221E-16
9.6	+1.298E-15	+1.998E-15
19.2	+1.499E-15	+5.637E-10

Results – Two-Phase Advection in Truchas

3D Shear Case





Conservation Error

New Transport (Youngs)

New Transport (Swartz)

Resolution	New (Swartz)	Maríc et. al
4.8	+3.315E-15	2.33E-15
7.2	-4.306E-15	-
9.6	+2.469E-15	4.91E-15
19.2	-1.504E-15	1.22E-14

Maríc et. al, JCP. 2018

Jofre et. al, C&F. 2014

Liovic et al, C&F. 2006

Results – Two-Phase Advection in Truchas

3D Deformation Case

Prescribed reversing velocity field

$$u(\vec{x},t) = \sin(2\pi y)\sin^2(\pi x)\sin(2\pi z)\cos(\pi t/3)$$
$$v(\vec{x},t) = -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos(\pi t/3)$$
$$w(\vec{x},t) = -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos(\pi t/3)$$

<u>Domain</u>

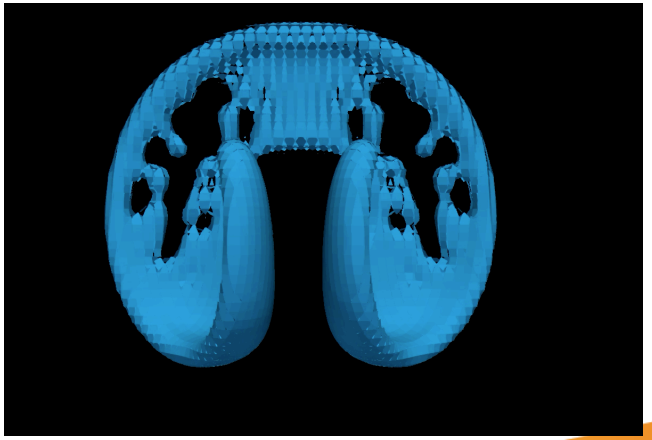
Dimensions $[0,0,0] \times [1,1,1]$

Discretized uniformly by $N \times N \times N$ hexahedron cells

Simulations performed at CFL of ~0.5

<u>Initial Shape</u>

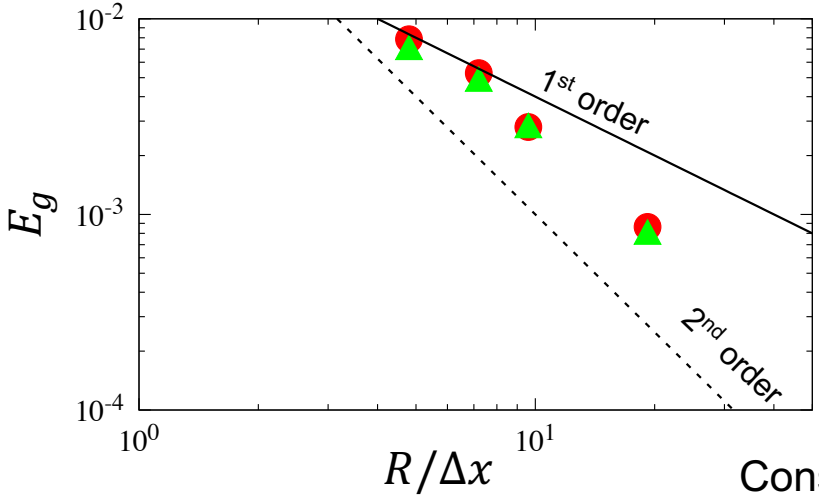
Sphere of radius 0.15, centered at $(0.35, 0.35, 0.35)$

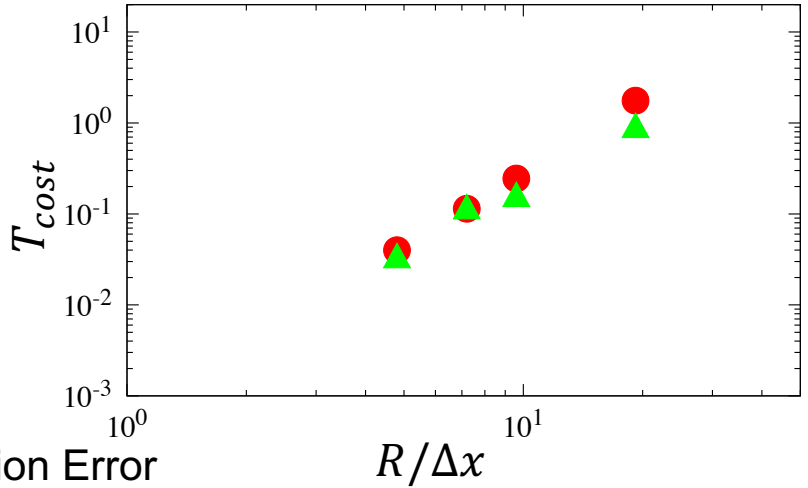


1. Enright et. al, JCP. 2002

Results – Two-Phase Advection in Truchas

3D Deformation Case - Hex





Conservation Error

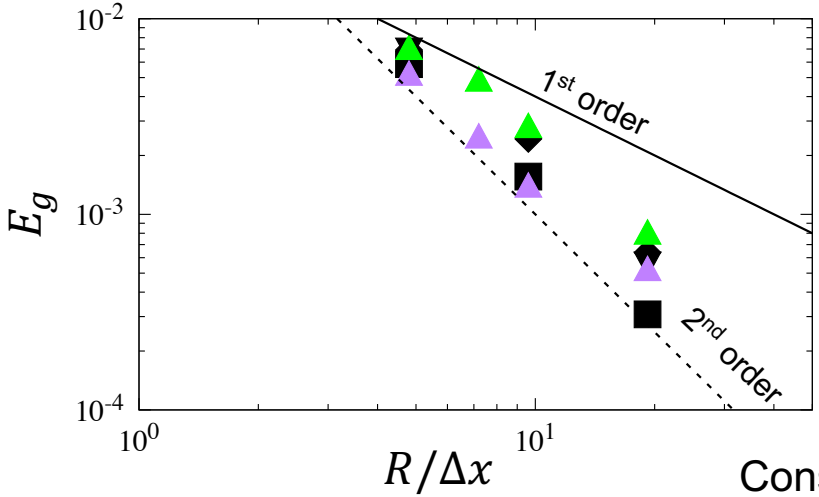
Old Method (Youngs)

New Transport (Youngs)

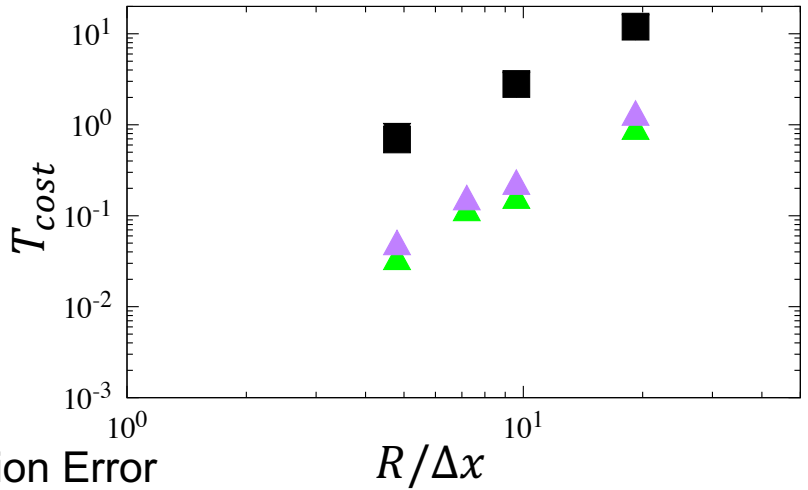
Resolution	New	Old
4.8	+4.077E-16	-2.342E-16
7.2	+5.135E-16	-2.949E-17
9.6	+8.795E-16	+2.307E-16
19.2	-6.811E-16	+2.841E-11

Results – Two-Phase Advection in Truchas

3D Deformation Case - Hex



Note: ▼ & ■ are coincident



▲ New Transport (Youngs)

▲ New Transport (Swartz)

Conservation Error

Resolution	New (Swartz)	Maríc et. al
4.8	-8.014E-16	2.46E-16
7.2	-3.296E-16	-
9.6	+2.307E-16	6.01E-15
19.2	+3.227E-16	1.56E-14

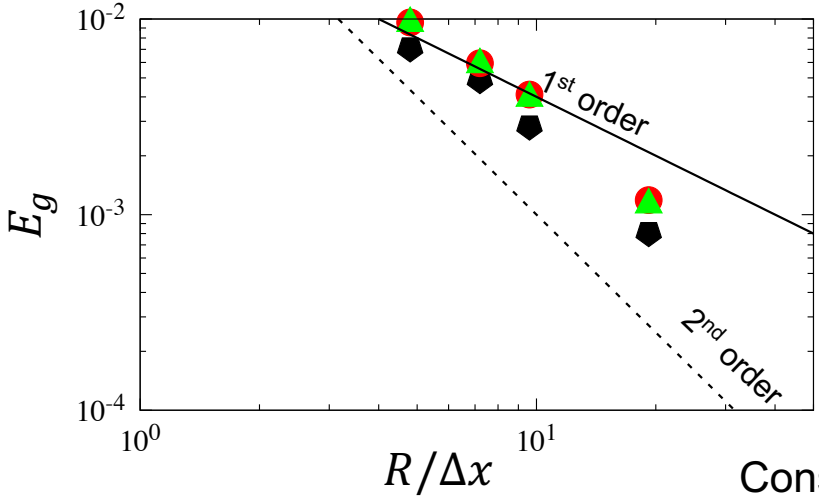
■ Maríc et. al, JCP. 2018

◆ Jofre et. al, C&F. 2014

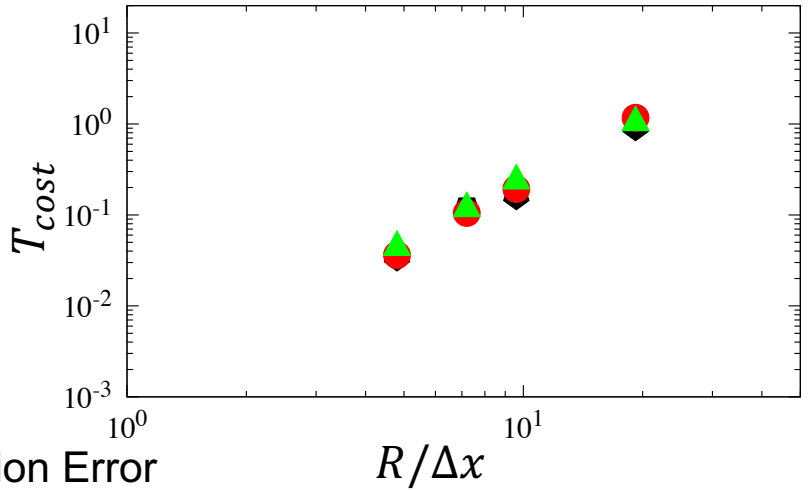
▼ Owkes & Desjardins, JCP. 2014

Results – Two-Phase Advection in Truchas

3D Deformation Case - Tet



Note: ▼ & ■ are coincident



● Old Method (Youngs)

▲ New Transport (Youngs)

Conservation Error

Resolution	New	Old
4.8	-2.018E-14	-4.539E-13
7.2	-8.342E-15	-4.023E-13
9.6	8.014E-16	-4.793E-13
19.2	-6.287E-13	-4.377E-09

⬟ New Transport (Youngs) on Hex

$$\Delta x_{tet}^3 \approx 6\sqrt{2}\,\Delta x_{hex}^3$$

Conclusions

IRL

- Library of computational geometry algorithms useful for unsplit VOF applications
- Robust, efficient, and general volume intersections routines
- Flexible voxelization framework for networks on convex volumes

Unsplit Advection in Truchas

- Efficient algorithm for efficient, discretely conservative geometric advection on unstructured meshes
- Imposes consistent face triangulation without centroid-based face decomposition
- Cell-wise advection and refinement of geometric advection band are two useful improvements to unsplit volume fraction methods

QUESTIONS?